Decision Problems for Propositional Linear Logic

Patrick Lincoln[1,3] *John Mitchell*[1] *Andre Scedrov*[2]
Natarajan Shankar[3]

September 13, 1994

[1]Department of Computer Science, Stanford University, Stanford, CA 94305
[2]Department of Mathematics, University of Pennsylvania, Philadelphia, PA 19104
[3]Computer Science Laboratory, SRI International, Menlo Park, CA 94025

Abstract

Linear logic, introduced by Girard, is a refinement of classical logic with a natural, intrinsic accounting of resources. This accounting is made possible by removing the "structural" rules of contraction and weakening; adding a modal operator; and adding finer versions of the propositional connectives. Linear logic has fundamental logical interest and applications to computer science, particularly to Petri nets, concurrency, storage allocation, garbage collection, and the control structure of logic programs. In addition, there is a direct correspondence between polynomial-time computation and proof normalization in a bounded form of linear logic.

In this paper we show that unlike most other propositional (quantifier-free) logics, full propositional linear logic is undecidable. Further, we prove that without the modal *storage* operator, which indicates unboundedness of resources, the decision problem becomes PSPACE-complete. We also establish membership in NP for the multiplicative fragment, NP-completeness for the multiplicative fragment extended with unrestricted weakening, and undecidability for fragments of noncommutative propositional linear logic.

Contents

1 Introduction **3**

2 Multiplicative Additive Propositional Linear Logic is PSPACE-complete **7**

2.1 Membership in PSPACE 8
2.2 Informal Outline of PSPACE-hardness of MALL 8
2.2.1 Encoding Boolean Evaluation 9
2.2.2 Encoding Boolean Quantification 12
2.3 Formal Definition of the Encoding 14
2.4 Proof of PSPACE-hardness of MALL 15

3 Propositional Linear Logic is Undecidable **23**

3.1 Overview 24
3.2 Linear Logic Augmented With Theories 24
3.3 Coding Theories in Formulas 26
3.4 And-Branching Two Counter Machines Without Zero-Test 28
3.4.1 Two Counter Machines 29
3.5 From Machines to Logic 31
3.6 Example Computation 37

4 Noncommutative Propositional Linear Logic **40**

4.1 NCL Proof Rules 41
4.2 NCL is Undecidable 41
4.3 NCL Theories 42
4.3.1 Semi-Thue Systems 43
4.4 From Semi-Thue Systems to Noncommutative Linear Logic 44
4.5 Other Noncommutative Logics 46
4.5.1 Rotate Rule versus Embedding 47
4.5.2 NCL without ?**E** 47
4.5.3 Alternate $\otimes$ 48
4.5.4 Mix and Match 48
4.6 Degenerate Noncommutative Linear Logics 49

4.6.1 Intermingling $\otimes$ 49
4.6.2 Intermingling **Cut** 49

5 The Multiplicative Fragment **51**
5.1 Multiplicatives 51
5.2 Direct Logic 51

6 Conclusion **56**

Bibliography **58**

A Cut Elimination **61**
A.1 Cut of non-principal formulas 63
A.1.1 $\otimes$ 63
A.1.2 $\parr$ 64
A.1.3 $\oplus$ 64
A.1.4 & 64
A.1.5 ?**W** 65
A.1.6 ?**C** 65
A.1.7 ?**D** 65
A.1.8 $\perp$ 66
A.1.9 $\top$ 66
A.1.10 **Cut** 66
A.2 Cut of principal formulas 66
A.2.1 I versus any 67
A.2.2 $\otimes$ **versus** $\parr$ 67
A.2.3 & **versus** $\oplus$ 67
A.2.4 ?**W** versus !**S** 67
A.2.5 ?**C** versus !**S** 68
A.2.6 ?**D** versus !**S** 68
A.2.7 !**S** versus !**S** 69
A.2.8 **Cut**$*$ versus !**S** 70
A.2.9 $\otimes$ versus !**S** 70
A.2.10 $\perp$ versus 1 71
A.3 Subformula Property 72

B Propositional Linear Logic Proof Rules **74**

Chapter 1

Introduction

Linear logic is a refinement of classical logic introduced by Girard [Gir87a]. This logic has a "resource sensitive" character, reflected in the fact that two assumptions of a formula A are distinguished from a single assumption of A. In this paper, we study the decision problem for full propositional linear logic and several natural fragments. The easiest result to state is that full propositional linear logic is undecidable. We also show that an "intuitionistic" fragment is undecidable, a natural fragment is PSPACE-complete, and a smaller fragment that is in NP becomes NP-complete with an additional rule. Before describing these results, we give a short overview of linear logic by explaining the passage from classical to linear logic.

Formally, linear logic may be derived in three steps from a Gentzen-style sequent calculus axiomatization of classical logic. The first step is to drop two structural rules, *contraction* and *weakening*. This forces a reexamination of conjunction and disjunction, leading to two forms of each connective. The third step is to recover the full expressive power of classical logic by adding two modal operators, ! and ?. These three steps are described in more detail in the following paragraphs. The resulting logic is surprisingly natural, from both proof-theoretic and computational standpoints. In particular, Gentzen-style *cut-elimination*, a crucial proof-theoretic property (see [Gen69, GLT89], for example), has been established for linear logic in [Gir87a]. This yields consistency and provides a natural computational mechanism that resembles reduction in lambda calculus (*e.g.*, [HS86, GLT89]).

The derivation of linear logic begins by dropping the structural rules *contraction* and *weakening,* which are an essential part of classical and intuitionistic logic. Each rule may be applied to either the left or right side of a sequent. On the left, contraction allows repeated assumptions of some formula to be replaced by a single assumption of the same formula. This means that a single hypothesis is as good as any number of duplicates, or, a hypothesis may be "reused" as often as desired. Contraction on the right also allows duplicates to be dropped, which has essentially the same effect. Weakening on the left allows us to add irrelevant hypotheses, and,

on the right, to extend the set of possible conclusions arbitrarily. Since contraction and weakening make it possible to use an assumption as little or as often as desired, these rules are responsible for what we may regard as a loss of control over resources in both classical and intuitionistic logic. Excluding these rules produces a *linear* system in which each assumption must be used *exactly once*, and each conclusion must follow from the hypotheses. In linear logic, formulas may be regarded as fixed resources that cannot necessarily be discarded or duplicated without effort.

The second step in deriving linear logic involves the propositional connectives. Briefly, the change in structural rules leads to two forms of conjunction and disjunction. The reason for the split is that we must decide whether we require linearity of the entire conjunction or disjunction, or whether it suffices to have each conjunct or disjunct alone depend linearly on the surrounding formulas. One form, called *additive* conjunction or disjunction, is informally described as "sharing of resources" since the two conjuncts or disjuncts may depend on a shared set of hypotheses. In the other *multiplicative* form, there is no such sharing. The general situation may be illustrated by examining two consequences of the pair of linear implications $A \multimap B$ and $A \multimap C$. Intuitively, $A \multimap B$ says that from A we may conclude B, or, in more computational terms, we have a process that will consume A and produce B. Given the two assumptions $A \multimap B$ and $A \multimap C$, there are two possible conclusions involving conjunctions of B and C. Using additive conjunction, written $\&$, we may conclude $A \multimap (B \,\&\, C)$ since from A we are capable of obtaining B and we are capable of obtaining C. With multiplicative conjunction, written $\otimes$, sharing is not allowed. However, we may obtain B and C from two separate A's. This is written symbolically as $(A \otimes A) \multimap (B \otimes C)$. One way of describing the distinction is that $B \,\&\, C$ indicates a choice between B and C, while $B \otimes C$ reflects an ability to have both simultaneously.

The final step in deriving linear logic is to add two modal operators. These are a *storage* or *reuse* operator, $!$, and a dual *consumption* operator $?$, definable from $!$ using negation. Intuitively, the formula $!A$ provides unlimited use of the resource A and $?B$ allows the unlimited consumption of B. Using a computational metaphor that we have found useful and faithful to the logic, we may read $!A$ as, "the datum A is stored in the memory and may be referenced an unlimited number of times". In deductive terms, if B follows from any number of assumptions of A, then B follows from the single assumption $!A$. A view of $!$ which suggests the translation of classical logic into linear logic is that while we do not have contraction and weakening as structural rules, we may apply contraction and weakening to formulas beginning with $!$. Since the basic framework remains linear, unbounded use is allowed only "locally", at formulas specifically marked with $!$ or $?$.

The first application of the resource-sensitive aspect of the logic was the development of a functional programming language implementation in which garbage collection was replaced by explicit duplication operations based on linear logic

[Laf88]. Further studies have demonstrated connections with Petri nets [Asp87, GG89, MOM89, AFG90, GG90] and other models of concurrency [Laf90, AV90]. With regard to concurrency, there is a similarity between *proof nets,* the inherent model of computation associated with cut-elimination in multiplicative linear logic (*cf* [Gir87a, Gir87b, DR89, Laf90]), and *connection graphs,* which were designed to model connection machine computation [Baw86]. Other applications include optimization of copying in lazy functional programming language implementation [GH90] and analyzing the control structure of logic programs [Cer90, AP90]. A natural characterization of polynomial time computations can be given in a bounded version of linear logic [GSS90] obtained by limiting reuse to specified bounds, *i.e.*, by bounding the number of references to each datum in memory. Informal introductions to linear logic may be found in [Gir90, Sce90].

We now summarize the main results of this paper (which were sketched in [LMSS90]), beginning with the smallest fragment considered. Multiplicative linear logic contains only linear implication, negation, and the multiplicative forms of conjunction and disjunction. Recall that these forms require the available resource to be partitioned rather than shared. We show that the decision problem for this fragment is in NP. With unrestricted weakening added, we show that the multiplicative fragment is NP-complete.

There are two natural fragments extending pure multiplicative linear logic, one with additive connectives and the other with the ! operator. We show that the first extension, with additive and multiplicative connectives but not !, is PSPACE-complete. The proof, by reduction from classical quantified boolean formulas, involves encoding quantifier order using only commutative propositional connectives. We note here in passing that the fragment with only multiplicative connectives and the ! operator is at least as hard as the reachability problem for Petri nets (or, equivalently, commutative semi-Thue systems or vector addition systems). This follows from conservativity properties established in this paper and previous work relating linear logic and Petri nets. Although reachability is decidable [May81, Kos82], the best known lower bound is EXPSPACE [Lip76, MM82]. A likely upper bound on Petri net reachability is primitive recursive in the Ackermann function [McA84, Clo86]. We do not know if multiplicative linear logic with ! is decidable.

Finally, we show that provability in full propositional linear logic with additive and multiplicative connectives and modal storage operator is undecidable. It follows from this undecidability result that when propositional linear logic is extended with quantification over propositions, the resulting logic is also undecidable. (Provability is trivially *recursively enumerable,* since the proof system is effective.) Undecidability also holds for a restricted form called intuitionistic propositional linear logic. In addition, we establish the undecidability of a noncommutative variant of linear logic (even without additive connectives), a system that extends the calculus in [Lam58]; see [Gir89, Yet90].

Our undecidability proof uses a direct encoding of a form of alternating counter machines. Our "and-branching counter machines" resemble the alternating Turing machines of [CKS81]), but lack a basic operation to test for zero. The basic transitions of these machines may be axiomatized using the multiplicative and additive connectives, while the ! operator is needed to allow an instruction to be executed an arbitrary number of times. Additive connectives are used to encode *and*-branching, which is needed to simulate the zero tests of conventional counter machines. As for the other lower bounds, the bulk of the technical work lies in establishing that the encoding is faithful, *i.e.*, each deduction in linear logic determines some computation. Faithfulness is demonstrated using a detailed examination of cut-elimination in linear logic [Gir87a]. This yields a version of the deduction theorem for linear logic and various conservativity results of independent interest.

A key insight is that searching for a proof of a certain special form for a given linear logic sequent corresponds directly to searching for an accepting computation in a particular machine model. A successful search is exactly an accepting computation.

For propositional linear logic without storage, membership in PSPACE is shown using a proof bound based on cut-elimination. PSPACE-hardness is demonstrated by a (log-space) construction of formulas that may be proved only by alternating between rules that simulate classical universal and existential propositional quantifiers. This construction demonstrates a surprising property of linear logic: the connectives are sufficient to express synchronization to the point of "sequentiality". Undecidability of noncommutative linear logic is proved by encoding the word problem for semigroups. Unlike our other reductions, this does not require the additive connectives. Membership in NP for multiplicative linear logic, with or without unrestricted weakening, is based on a polynomial bound on proof-size resulting from cut-elimination. With unrestricted weakening, we show NP-completeness by reduction from the vertex cover problem [GJ79].

A logic that is superficially related to linear logic is propositional relevance logic, which is proved undecidable in [Urq84]. Like linear logic, relevance logic lacks weakening. However, relevance logic *does* have unrestricted contraction. In addition, relevance logic has a distributivity axiom, absent from linear logic. Without the distributivity axiom, relevance logic becomes decidable [Mey66]. The system with distributivity also lacks cut-free Gentzen-type formulation. See, for example, [Avr88]. Thus both the motivation and technical properties of linear logic are significantly different from relevance logic.

Chapter 2

Multiplicative Additive Propositional Linear Logic is PSPACE-complete

In this chapter, we analyze the complexity of the fragment of propositional linear logic without the modal storage operator ! and its dual ?, but including all the remaining connectives and constants of linear logic.

We begin with some standard definitions. A *deduction* in propositional linear logic is a tree, usually presented with the root at the bottom, and the leaves at the top. Each branch of a deduction is a sequence of applications of the proof rules given in Appendix B, some of which, such as $\&$, represent branching points in the deduction tree, some, such as $\parr$, which extend the length of a branch, and some, such as identity, which terminate a branch. The leaves embody the assumptions, and the root the conclusion. Such a structure is said to be a deduction of the conclusion from the assumptions. A *proof* in linear logic is a deduction with no assumptions. That is, each branch terminates with an application of identity, $\top$, or 1. One interesting feature of linear logic, as presented in Appendix B, is that negation is defined, and it is not a connective. In particular, the propositional literals are assumed to be given in pairs, one positive (written p_i for some i) and one negative (written $p_i^{\perp}$).

In this chapter we are concerned with the multiplicative-additive fragment of linear logic, which we abbreviate as MALL. The logical symbols used in this fragment are multiplicative conjunction ($\otimes$) and disjunction ($\parr$), additive conjunction ($\&$) and disjunction ($\oplus$), and the constants 0, 1, $\top$, and $\perp$. MALL formulas and sequents contain only these connectives and constants, in addition to the positive and negative literals. The proof rules of MALL are all of the rules in the Appendix B that are associated with these connectives and constants. This logic has been studied

by Girard [Gir87a] and Bellin [Bel90]. While provability for the classical propositional logic is co-NP-complete, we show below that provability for MALL is PSPACE-complete.

An important property of the sequent calculus formulation of MALL is cut-elimination. This property follows from the Theorem A.3 of Appendix A.

Theorem 2.1 *Any sequent provable in* MALL *is provable without the cut rule.*

Proof. Since MALL is a fragment of linear logic, we may use the cut-elimination procedure from Theorem A.3 to convert a MALL proof to a cut-free proof in linear logic. By the subformula property (Theorem A.4), such a cut-free proof of a MALL sequent contains only MALL formulas. Since all the rules which apply to MALL formulas are already in MALL, any cut-free proof of a MALL sequent must already be a MALL proof. ∎

Membership in PSPACE is straightforward, given cut elimination, but we include a short sketch to illustrate the importance of Theorem 2.1. The proof of PSPACE-hardness is more technical. Proof search in the cut-free sequent calculus is crucial to the proof. The primitive step in proof search is a *reduction*, namely the application of an inference rule to transform a sequent matching the conclusion of the rule to the collection of sequents given by the corresponding premises of the rule. A reduction is the inverse of an inference rule, and drives conclusions to premises. Proof search is the process of constructing a cut-free proof in a bottom-up manner by nondeterministically applying reductions starting from the conclusion sequent.

2.1 Membership in PSPACE

Theorem 2.2 *The provability in* MALL *of a given sequent can be decided by a polynomial-space bounded Turing machine.*

Proof. By Theorem 2.1, a provable MALL sequent has a cut-free MALL proof. In a cut-free MALL proof, there are at most two premises to each rule, and each premise is strictly smaller than the consequent. Therefore, the depth of a cut-free MALL proof tree is at most linear in the length of the final sequent of the proof. An alternating Turing machine [CKS81] may guess and check a cut-free proof in linear time, using OR-branching to nondeterministically guess a reduction in the cut-free proof, and AND-branching to generate and check the proofs of both premises of a two premise rule in parallel.

Membership in PSPACE can also be proved without reference to alternation. A nondeterministic Turing machine can be defined to generate and check a cut-free sequent proof in a depth-first manner. Given the linear bound on the depth of any cut-free proof with respect to the size of the conclusion sequent, the search stack

need contain no more than a linear number of sequents. Since each sequent in a cut-free proof is no larger than the conclusion sequent, we get a quadratic bound on the stack size. ∎

2.2 Informal Outline of PSPACE-hardness of MALL

Since there are a number of technical details to the proof of PSPACE-hardness, we will illustrate the key intuitions by means of an example; the details of the proof are given in Section 2.4.

The PSPACE-hardness of MALL provability is demonstrated by a transformation from the validity problem for quantified Boolean formulas (QBF). A *quantified Boolean formula* has the (prenex) form $Q_m X_m \ldots Q_1 X_1 \colon M$, where

1. each Q_i is either $\forall$ or $\exists$,
2. M is a quantifier-free *Boolean matrix* containing only the connectives $\neg$ and $\wedge$, and *Boolean variables*.

A *closed* QBF contains no free variables. Our conventions in this section are that G and H range over quantified Boolean formulas; M and N range over quantifier-free Boolean formulas; U, V, X, Y, Z range over Boolean variables; and I ranges over truth value assignments. For expository convenience, we refer to quantifier-free Boolean formulas simply as Boolean formulas.

An *assignment* I for a set of Boolean variables $\{X_1, \ldots, X_n\}$ maps each X_i to a truth value from $\{T, F\}$. An assignment is represented by a sequence of Boolean variables and negated Boolean variables. For example, the assignment $X_1, \neg X_2, X_3$ maps X_1 to T, X_2 to F, and X_3 to T. The assignment I, X assigns T to X, but behaves like I, otherwise. If I is an assignment for the free variables in G, we use the standard notation $I \models G$ to indicate that G is valid under I, and write $I \not\models G$ if I falsifies G. Note that

$$I \models \forall X \colon G \text{ iff } I, X \models G \quad \text{and} \quad I, \neg X \models G$$
$$I \models \exists X \colon G \text{ iff } I, X \models G \quad \text{or} \quad I, \neg X \models G$$

If G is a QBF and I is an assignment for the free variables in G, we say G *is valid under* I exactly if $I \models G$. If G is a closed QBF, then G is said to be valid if it is valid under the empty assignment. The validity of a closed QBF G is represented as $\models G$. The QBF validity problem is: *Given a closed QBF G, is G valid?*

We demonstrate the PSPACE-hardness of MALL provability by defining a succinct encoding of a QBF as a MALL sequent that is provable *exactly* when the given QBF is valid.

The transformation of the QBF validity problem to MALL provability takes place in two steps:

- Given a quantifier-free Boolean formula M and an assignment I for the free variables in M, we show that there is a MALL sequent encoding M and I which is provable exactly when M is valid under I. This essentially demonstrates that the process of evaluating Boolean functions can be represented by the process of cut-free proof search in the MALL sequent calculus.
- Given a QBF G and an assignment I for the free variables in G, there exists a MALL sequent encoding the quantifier prefix and the Boolean matrix of G so that the MALL sequent is provable exactly when G is valid under I. The idea here is to simulate the Boolean quantifiers $\exists$ and $\forall$ by using the additive connectives $\oplus$ and $\&$.

Two-sided vs. one-sided sequents. We use a formulation of MALL with one-sided sequents to simplify the proofs. In linear logic, a two-sided sequent $A_1, \ldots, A_m \vdash B_1, \ldots, B_n$ has the one-sided form $\vdash A_1^\perp, \ldots, A_m^\perp, B_1, \ldots, B_n$. Thus, a formula $A \multimap B$ on the left of a two-sided sequent becomes $A \otimes B^\perp$ in a one-sided sequent. Similarly, the provable two-sided sequent $A, A \multimap B \vdash B$ becomes $\vdash A^\perp, A \otimes B^\perp, B$. While one-sided sequents simplify the technical arguments considerably, the reader might gain further insight by rewriting parts of our encoding in a two-sided form.

2.2.1 Encoding Boolean Evaluation

The encoding of the Boolean connectives and quantifiers in MALL is described here by means of an example. The full definition of the encoding appears in Section 2.3. The encoding from QBF validity to MALL provability makes no use of the MALL constants. Consider the *valid* QBF G given by

$$\forall X_2 \exists X_1 \colon \neg(\neg X_1 \wedge X_2) \wedge \neg(\neg X_2 \wedge X_1).$$

The matrix M of G is essentially a restatement of $(X_1 \iff X_2)$. Let H be the falsifiable formula $\exists X_1 \forall X_2 \colon M$ that is obtained from G by reversing the order of the quantifiers. It is crucial that the encodings of G and H in MALL respect the ordering of quantifiers so that the encoding of G is provable but the encoding of H is not.

The encoding of the Boolean matrix describes the formula as a circuit with signals labeled by MALL literals. Let the assignment I be encoded by a sequence of MALL formulas $\langle I \rangle$, and $[M]_a$ be the MALL formula encoding M with output labeled by the literal a. Then $I \models M$ is encoded by the sequent

$$\vdash \langle I \rangle, [M]_a, a$$

whereas $I \not\models M$ is encoded by

$$\vdash \langle I \rangle, [M]_a, a^\perp.$$

Since we are using one-sided sequents, we encode the assignment $X_1, \neg X_2$ by $x_1^\perp, x_2$. The MALL literals encoding the assignment are to be seen as the input signals to the encoding of the Boolean formula.

We first consider the Boolean connectives $\neg$ and $\wedge$, then construct the full encoding of M. The encoding $[\neg X_1]_a$ of $\neg X_1$ with output labeled a is the formula $\text{NOT}(x_1, a)$. For literals x and y, the definition of $\text{NOT}(x, y)$ is just the representation of the truth table for negation within MALL, as shown below:

$$\text{NOT}(x, y) = (x \otimes y) \oplus (x^\perp \otimes y^\perp). \tag{2.1}$$

$\text{NOT}(x_1, a)$ is simply the linear negation of the formula

$$(x_1 \multimap a^\perp) \& (x_1^\perp \multimap a)$$

which is more perspicuous in describing a as the Boolean negation of x_1. The sequent

$$\vdash x_1, \text{NOT}(x_1, a), a \tag{2.2}$$

encodes the situation where the input X_1 is F, and asserts (correctly) that the output $\neg X_1$ is T.

The sequent (2.2) is easily seen to have the MALLproof

$$\dfrac{\dfrac{\dfrac{}{\vdash x_1, x_1^\perp}\mathbf{I} \qquad \dfrac{}{\vdash a^\perp, a}\mathbf{I}}{\vdash x_1, (x_1^\perp \otimes a^\perp), a}\otimes}{\vdash x_1, (x_1 \otimes a) \oplus (x_1^\perp \otimes a^\perp), a}\oplus$$

Similarly, the sequent (2.3) representing $\{X_1 \leftarrow T\} \not\models \neg X_1$ is also provable.

$$\vdash x_1^\perp, \text{NOT}(x_1, a), a^\perp. \tag{2.3}$$

On the other hand, the sequent

$$\vdash x_1^\perp, \text{NOT}(x_1, a), a \tag{2.4}$$

asserts (falsely) that $\{X_1 \leftarrow T\} \models \neg X_1$. To see why sequent (2.4) is not provable, we observe that MALL is a refinement of classical logic in which no classically falsifiable sequents are provable. The sequent $\vdash x_1^\perp, \text{NOT}(x_1, a), a$ is falsified by assigning T to x_1 and F to a, while interpreting $\otimes$ and $\&$ as classical conjunction and $\oplus$ and $\parr$ as classical disjunction. A sequent is interpreted classically as the disjunction of the sequence of formulas that it contains.

The encoding for conjunction, $[X \wedge Y]_b$ is given by $\text{AND}(x, y, b)$ as defined below.

$$\textsc{and}(x,y,b) = \begin{bmatrix} (x \otimes y \otimes b^{\perp}) & \oplus \\ (x^{\perp} \otimes y^{\perp} \otimes b) & \oplus \\ (x \otimes y^{\perp} \otimes b) & \oplus \\ (x^{\perp} \otimes y \otimes b) & \end{bmatrix} \tag{2.5}$$

Sequent (2.6) represents $X, Y \models (X \wedge Y)$:

$$\vdash x^{\perp}, y^{\perp}, \textsc{and}(x,y,b), b. \tag{2.6}$$

Sequent (2.6) has the proof

$$\dfrac{\dfrac{\dfrac{}{\vdash x^{\perp}, x}\mathbf{I} \qquad \dfrac{\dfrac{}{\vdash y^{\perp}, y}\mathbf{I} \qquad \dfrac{}{\vdash b^{\perp}, b}\mathbf{I}}{\vdash y^{\perp}, (y \otimes b^{\perp}), b}\otimes}{\vdash x^{\perp}, y^{\perp}, (x \otimes y \otimes b^{\perp}), b}\otimes}{\vdash x^{\perp}, y^{\perp}, \textsc{and}(x,y,b), b}\oplus$$

As with sequent (2.4), the MALL sequent representing the false assertion $\neg X, Y \models (X \wedge Y)$ is given by

$$\vdash x, y^{\perp}, \textsc{and}(x,y,b), b$$

and is not provable since it can be falsified by the classical interpretation assigning F to x and b, and T to y.

The next step is to construct the encoding of the Boolean formula M given at the beginning of this section, from the encodings of the Boolean connectives. The formula M is thought of as a Boolean circuit with the distinctly labeled signals. The encoding $[(\neg X_1 \wedge X_2)]_b$ is given by the formula $\textsc{and}(a, x_2, b) \parr \textsc{not}(x_1, a)$. Let $\textsc{implies}(x,y,u,v,w)$ represent the formula

$$\textsc{not}(v,w) \parr \textsc{and}(u,y,v) \parr \textsc{not}(x,u),$$

then $\textsc{implies}(x_1, x_2, a, b, c)$ is the encoding $[\neg(\neg X_1 \wedge X_2)]_c$. The literals a, b and c are the distinct literals labeling the output signals of the Boolean gates.

We now consider the problem that the input signals in M have a fanout greater than one. An almost correct encoding in MALL of the Boolean formula M is given by the formula

$$\textsc{and}(c,f,g) \parr \textsc{implies}(x_1,x_2,a,b,c) \parr \textsc{implies}(x_2,x_1,d,e,f).$$

The validity of M under the assignment $\{X_1 \leftarrow T, X_2 \leftarrow T\}$ would then be represented by

$$\vdash x_1^{\perp}, x_2^{\perp}, \textsc{and}(c,f,g) \parr \textsc{implies}(x_1,x_2,a,b,c) \parr \textsc{implies}(x_2,x_1,d,e,f), g. \tag{2.7}$$

The following deduction represents one attempt to prove sequent (2.7).

$$
\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash g^{\perp}, g}\mathbf{I} \qquad \dfrac{\begin{array}{c}\vdots\\ \vdash x_1^{\perp}, x_2^{\perp}, \textsc{implies}(x_1, x_2, a, b, c), c\end{array} \qquad \begin{array}{c}\vdots\\ \vdash \textsc{implies}(x_2, x_1, d, e, f), f\end{array}}{\vdash x_1^{\perp}, x_2^{\perp}, (c \otimes f), \textsc{implies}(x_1, x_2, a, b, c), \textsc{implies}(x_2, x_1, d, e, f)}\otimes}{\vdash x_1^{\perp}, x_2^{\perp}, (c \otimes f \otimes g^{\perp}), \textsc{implies}(x_1, x_2, a, b, c), \textsc{implies}(x_2, x_1, d, e, f), g}\otimes}{\vdash x_1^{\perp}, x_2^{\perp}, \textsc{and}(c, f, g), \textsc{implies}(x_1, x_2, a, b, c), \textsc{implies}(x_2, x_1, d, e, f), g}\oplus}{\vdash x_1^{\perp}, x_2^{\perp}, \textsc{and}(c, f, g) \parr \textsc{implies}(x_1, x_2, a, b, c) \parr \textsc{implies}(x_2, x_1, d, e, f), g}\parr
$$

Since MALL lacks a rule of contraction, each of the assignment literals, $x_1^{\perp}$ and $x_2^{\perp}$, can appear in only one premise of a $\otimes$ rule. As a result, one of the remaining subgoals in the above deduction lacks the required input literals. We therefore need to be able to explicitly duplicate the assignment literals in the sequent (2.7) to match the number of duplicate occurrences of X_1 and X_2 in M. The formula $\textsc{copy}(x_1)$ defined as

$$(x_1 \otimes (x_1^{\perp} \parr x_1^{\perp})) \oplus (x_1^{\perp} \otimes (x_1 \parr x_1))$$

serves to duplicate an instance of x_1 or $x_1^{\perp}$. If M is now encoded as

$$\textsc{and}(c, f, g) \parr \textsc{copy}(x_1) \parr \textsc{copy}(x_2) \parr \Gamma_1 \parr \Gamma_2$$

where Γ_1 abbreviates $\textsc{implies}(x_1, x_2, a, b, c)$, and Γ_2 abbreviates $\textsc{implies}(x_2, x_1, d, e, f)$, the desired deduction of (2.7) can then be constructed.

$$
\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash x_1, x_1^{\perp}}\mathbf{I} \qquad \dfrac{\dfrac{}{\vdash x_2, x_2^{\perp}}\mathbf{I} \qquad \dfrac{\dfrac{}{\vdash g, g^{\perp}}\mathbf{I} \qquad \dfrac{\begin{array}{c}\vdots\\ \vdash x_1^{\perp}, x_2^{\perp}, \Gamma_1, c\end{array} \qquad \begin{array}{c}\vdots\\ \vdash x_1^{\perp}, x_2^{\perp}, \Gamma_2, f\end{array}}{\vdash x_1^{\perp}, x_1^{\perp}, x_2^{\perp}, x_2^{\perp}, (c \otimes f), \Gamma_1, \Gamma_2}\otimes}{\vdash x_1^{\perp}, x_1^{\perp}, x_2^{\perp}, x_2^{\perp}, (c \otimes f \otimes g^{\perp}), \Gamma_1, \Gamma_2, g}\otimes}{\vdash x_1^{\perp}, x_1^{\perp}, x_2^{\perp}, \textsc{and}(c, f, g), x_2 \otimes (x_2^{\perp} \parr x_2^{\perp}), \Gamma_1, \Gamma_2, g}\otimes}{\vdash x_1^{\perp}, x_2^{\perp}, \textsc{and}(c, f, g), x_1 \otimes (x_1^{\perp} \parr x_1^{\perp}), \textsc{copy}(x_2), \Gamma_1, \Gamma_2, g}\otimes}{\vdash x_1^{\perp}, x_2^{\perp}, \textsc{and}(c, f, g), \textsc{copy}(x_1), \textsc{copy}(x_2), \Gamma_1, \Gamma_2, g}\oplus}{\vdash x_1^{\perp}, x_2^{\perp}, \textsc{and}(c, f, g) \parr \textsc{copy}(x_1) \parr \textsc{copy}(x_2) \parr \Gamma_1 \parr \Gamma_2, g}\parr
$$

In summary, we have informally described the encoding in MALL of the evaluation of Boolean formulas under an assignment. The connectives $\parr$, $\otimes$, and $\oplus$ were used to represent the truth tables of $\neg$ and $\wedge$, and MALL literals were used to represent the "signals" in the Boolean formula. The duplication of input signals forms a crucial part of the encoding since MALL lacks a rule of contraction.

2.2.2 Encoding Boolean Quantification

Recall that G is the formula $\forall X_2 \exists X_1 \colon M$, and H is the formula $\exists X_1 \forall X_2 \colon M$, where M is $\neg(\neg X_1 \wedge X_2) \wedge \neg(\neg X_2 \wedge X_1)$. Intuitively, it is useful to separate the encoding of the Boolean quantifier prefix as separately encoding the individual quantifiers and the dependencies between quantifiers. Given the above encoding for assignments and Boolean formulas, an almost correct way to encode Boolean quantifiers would be to encode $\exists X_1$ as the formula $(x_1 \oplus x_1^\perp)$, and $\forall X_2$ as $(x_2 \& x_2^\perp)$. The encoding of G would then be given by the sequent

$$\vdash (x_2 \& x_2^\perp), (x_1 \oplus x_1^\perp), [M]_g, g.$$

The formula $(x_1 \oplus x_1^\perp)$ behaves like existential quantification in proof search since a nondeterministic choice can be made between

$$\dfrac{\begin{array}{c}\vdots \\ \vdash x_1^\perp, \Gamma\end{array}}{\vdash (x_1 \oplus x_1^\perp), \Gamma}\oplus \qquad \text{and} \qquad \dfrac{\begin{array}{c}\vdots \\ \vdash x_1, \Gamma\end{array}}{\vdash (x_1 \oplus x_1^\perp), \Gamma}\oplus$$

according to the assignment (T or F, respectively) to X_1 which makes $\exists X_1 \colon M$ valid. Similarly, the rule for reducing $(x_2 \& x_2^\perp)$ in a proof behaves like universal quantification requiring proofs of both $\vdash x_2^\perp, \Gamma$ and $\vdash x_2, \Gamma$.

$$\dfrac{\begin{array}{c}\vdots \\ \vdash x_2^\perp, \Gamma\end{array} \qquad \begin{array}{c}\vdots \\ \vdash x_2, \Gamma\end{array}}{\vdash (x_2 \& x_2^\perp), \Gamma}\&$$

However, with this mapping of quantifiers, the MALL encoding of G and H would be identical and provable, but H is not a valid QBF.

A correct encoding of $\exists X_1 \forall X_2 \colon M$ should ensure that if the encoding is provable in MALL, then there is a proof in which the choice of a truth value for X_1 is independent of whether X_2 is T or F. The order of reductions below show how the choice of a truth value for $\exists X_1$ in a proof of the MALL encoding can depend on the quantifier $\forall X_2$.

$$\dfrac{\dfrac{\begin{array}{c}\vdots \\ \vdash x_1, x_2, \Gamma\end{array}}{\vdash (x_1 \oplus x_1^\perp), x_2, \Gamma}\oplus \qquad \dfrac{\begin{array}{c}\vdots \\ \vdash x_1^\perp, x_2^\perp, \Gamma\end{array}}{\vdash (x_1 \oplus x_1^\perp), x_2^\perp, \Gamma}\oplus}{\vdash (x_1 \oplus x_1^\perp), (x_2 \& x_2^\perp), \Gamma}\&$$

In this ordering of the reductions, $(x_1 \oplus x_1^\perp)$ is reduced differently on the x_2 and $x_2^\perp$ branches of the proof leading to distinct witnesses for X_1 according to whether

X_2 is T or F. The solution to this quantifier order problem is to encode the quantifier dependencies in the MALL formula so that if there is any proof, then there is some proof of the encoding in which $(x_1 \oplus x_1^\perp)$ is reduced *below* $(x_2 \& x_2^\perp)$, thus ensuring that the truth value of X_1 has been chosen independently of the truth value for X_2. For this purpose, we introduce new MALL atoms q_0, q_1, q_2, and encode $\exists X_1 \forall X_2 \colon M$ as

$$\begin{array}{ll} \vdash & q_2, \\ & q_2^\perp \otimes ((q_1 \parr x_1) \oplus (q_1 \parr x_1^\perp)), \\ & q_1^\perp \otimes ((q_0 \parr x_2) \& (q_0 \parr x_2^\perp)), \\ & q_0^\perp \otimes [M]_g, g \end{array}$$

The idea here is that the quantifier encoding for $\exists X_1$ hides the "key" q_1 that is needed to unlock the quantifier encoding for $\forall X_2$. If we now attempt to violate the quantifier dependencies, the following would be one possible deduction.

$$\dfrac{\dfrac{\dfrac{}{\vdash q_2, q_2^\perp}\mathbf{I} \qquad \dfrac{\dfrac{\dfrac{?}{\vdash q_1^\perp, q_1, x_1}}{\vdash q_1^\perp, q_1 \parr x_1}\parr}{\vdash q_1^\perp, ((q_1 \parr x_1) \oplus (q_1 \parr x_1^\perp))}\oplus}{\vdash q_2, q_1^\perp, q_2^\perp \otimes ((q_1 \parr x_1) \oplus (q_1 \parr x_1^\perp))}\otimes \qquad \begin{array}{c}\vdots \\ \vdash (q_0 \parr x_2) \& (q_0 \parr x_2^\perp), q_0^\perp \otimes [M]_g, g\end{array}}{\vdash q_2, q_2^\perp \otimes ((q_1 \parr x_1) \oplus (q_1 \parr x_1^\perp)), q_1^\perp \otimes ((q_0 \parr x_2) \& (q_0 \parr x_2^\perp)), q_0^\perp \otimes [M]_g, g}\otimes$$

In the above deduction, we are left with a subgoal of the form $\vdash q_1^\perp, q_1, x_1$, and since x_1 is not a constant, we cannot reduce this sequent to a MALL axiom. (Recall that MALL lacks an unrestricted weakening rule.) Other deductions attempting to violate the quantifier ordering also fail. On the other hand, the deduction which does respect the order of the quantifier encodings can be performed as shown below. The quantifier encoding for $\exists X_1$ provides the key q_1 for unlocking the quantifier encoding of $\forall X_2$.

$$\dfrac{\dfrac{\dfrac{}{\vdash q_2, q_2^\perp}\mathbf{I} \qquad \dfrac{\dfrac{\begin{array}{c}\vdots \\ \vdash q_1, x_1, q_1^\perp \otimes ((q_0 \parr x_2) \& (q_0 \parr x_2^\perp)), q_0^\perp \otimes [M]_g, g\end{array}}{\vdash q_1 \parr x_1, q_1^\perp \otimes ((q_0 \parr x_2) \& (q_0 \parr x_2^\perp)), q_0^\perp \otimes [M]_g, g}\parr}{\vdash ((q_1 \parr x_1) \oplus (q_1 \parr x_1^\perp)), q_1^\perp \otimes ((q_0 \parr x_2) \& (q_0 \parr x_2^\perp)), q_0^\perp \otimes [M]_g, g}\oplus}{\vdash q_2, q_2^\perp \otimes ((q_1 \parr x_1) \oplus (q_1 \parr x_1^\perp)), q_1^\perp \otimes ((q_0 \parr x_2) \& (q_0 \parr x_2^\perp)), q_0^\perp \otimes [M]_g, g}\otimes$$

The formal definition of the polynomial time encoding of QBF validity in terms of MALL provability is given in Section 2.3. In Section 2.4, we demonstrate the correctness of the encoding.

2.3 Formal Definition of the Encoding

For our purpose, a Boolean formula is constructed from Boolean variables using the Boolean connectives $\neg$ and $\wedge$. All quantified variables are assumed to occur in the matrix. $Var(M)$ is the set of variables occurring in the Boolean formula M. Overlined syntactic variables such as $\overline{X}$ and $\overline{Y}$ range over sets of Boolean variables.

The mall sequent encoding a QBF G is represented by $\sigma(G)$. We need to be careful about keeping literals distinct. The annotation "a new" in the definition indicates that the literal a is a freshly chosen one that has not been used elsewhere in the encoding.

The sequent $\sigma(G)$ consists of the encoding of the QBF $[\![G]\!]_g$, where g labels the output signal, the key q_n, and the output value g. The definition of $[\![G]\!]_g$ constructs the quantifier encodings by induction on the length of the quantifier prefix. The definition of $[M]_g$ is by induction on the structure of M, so that $[N \wedge P]_g$ is constructed by

- choosing the fresh labels a and b for the outputs of subformulas N and P, respectively
- defining the relation between a, b, and g by $\textsc{and}(a,b,g)$
- if needed, providing a copying formula for each Boolean variable common to both N and P
- and recursively constructing $[N]_a$ and $[P]_b$

To be precise, we provide the following definition of the encoding.

Definition 2.3

$$
\begin{array}{rcll}
\sigma(G) & = & \vdash q_n, [\![G]\!]_g, g & q_n, g \ \textit{new} \\
[\![(\forall X_{i+1}{:}G)]\!]_g & = & (q_{i+1}^{\perp} \otimes ((x_{i+1} \parr q_i) \& (x_{i+1}^{\perp} \parr q_i))), [\![G]\!]_g & q_{i+1} \ \textit{new} \\
[\![(\exists X_{i+1}{:}G)]\!]_g & = & (q_{i+1}^{\perp} \otimes ((x_{i+1} \parr q_i) \oplus (x_{i+1}^{\perp} \parr q_i))), [\![G]\!]_g & q_{i+1} \ \textit{new} \\
[\![M]\!]_g & = & (q_0^{\perp} \otimes [M]_g) & q_0 \ \textit{new} \\
[X]_g & = & (x^{\perp} \otimes g) \oplus (x \otimes g^{\perp}) & \\
[\neg N]_g & = & \textsc{not}(a,g) \parr [N]_a & a \ \textit{new} \\
[N \wedge P]_g & = & \left\{ \begin{array}{r} \textsc{and}(a,b,g) \ \parr \\ \textsc{copyall}(Var(N) \cap Var(P)) \ \parr \\ [N]_a \ \parr \\ [P]_b \end{array} \right. & a, b \ \textit{new}; \ Var(N) \cap Var(P) \neq \emptyset \\
[N \wedge P]_g & = & \textsc{and}(a,b,g) \parr [N]_a \parr [P]_b & a, b \ \textit{new}; \ Var(N) \cap Var(P) = \emptyset \\
\textsc{copyall}(\overline{X}) & = & \parr_{X_i \in \overline{X}} \textsc{copy}(x_i) &
\end{array}
$$

Note that the sequent $\sigma(G)$ contains no MALL constants. The complexity of computing $\sigma(G)$ is at most quadratic in the size of G since the encoding function is defined inductively over the structure of the formula, and the intersection operation can be performed in linear time with a bit-vector representation of sets, where the length of each bit-vector is the number of distinct Boolean variables occurring in G. The cost of constructing the COPY formulas at each step in the recursion is also linear in the size of G. The cost of each NOT and AND formula is fixed with respect to the representation of the literals, and the literals can be represented with a cost that is logarithmic in the size of G.

The encoding may be computed in log-space, although the algorithm described above uses more than log-space, because of the work space required to save the set of variables that must be copied when encoding a conjunction. The encoding algorithm could be modified to make a number of passes over the input to determine the number of occurrences of each variable and generate the required number of COPY formulas. Each pass would use only log-space, and the remainder of the algorithm may be performed in log-space.

2.4 Proof of PSPACE-hardness of MALL

The main theorem is that for any closed QBF G, G is valid if and only if $\sigma(G)$ is a provable MALL sequent. The first set of lemmas demonstrates that the encoding of Boolean formulas works correctly. The second set of lemmas demonstrates that the Boolean quantifiers have been correctly encoded.

If I is a truth value assignment for the Boolean variables $X_1, \ldots, X_n$, then I is encoded as $\langle I \rangle$, where

$$\begin{array}{rcl} \langle I \rangle & = & \langle X_1 \rangle_I, \ldots, \langle X_n \rangle_I \\ \langle X_i \rangle_I & = & \begin{cases} x_i^{\perp} & \text{if } I(X_i) = T \\ x_i, \text{ otherwise} & \end{cases} \end{array}$$

If I is an assignment for a set of variables $\overline{Y}$, and $\overline{X} \subseteq \overline{Y}$, then $I/\overline{X}$ is the assignment I restricted to the subset $\overline{X}$, and by abuse of notation I/M is $I/Var(M)$. The following lemma is stated without proof.

Lemma 2.4 *Given sets of variables $\overline{X}$ and $\overline{Y}$, and an assignment I for $\overline{X} \cup \overline{Y}$, there is a deduction of the sequent $\vdash \langle I \rangle, \text{COPYALL}(\overline{X} \cap \overline{Y}), \Gamma$ from the sequent $\vdash \langle I/\overline{X} \rangle, \langle I/\overline{Y} \rangle, \Gamma$.*

Lemma 2.5 *Let M be a Boolean formula and I an assignment for the variables in M, then*

1. *if $I \models M$ then $\vdash \langle I \rangle, [M]_g, g$*

2. *if* $I \not\models M$ *then* $\vdash \langle I \rangle, [M]_g, g^\perp$

Proof. By induction on the structure of M, as follows. The cases in the proof correspond closely to those in the definition of $[M]_g$.

Base case: $M \equiv X$. Suppose $I(X) = T$, then $I \models M$ and $\langle I \rangle = x^\perp$. The following proof can then be constructed, expanding the definition of $[M]_g$

$$\dfrac{\dfrac{\dfrac{}{\vdash x^\perp, x}\mathbf{I} \qquad \dfrac{}{\vdash g^\perp, g}\mathbf{I}}{\vdash x^\perp, (x \otimes g^\perp), g}\otimes}{\vdash x^\perp, (x^\perp \otimes g) \oplus (x \otimes g^\perp), g}\oplus$$

The case when $I(X) = F$ is similarly straightforward.

Induction step: There are a number of cases here corresponding to the definition of $[M]_g$. We consider a typical case and leave the remaining ones to the reader.

Let $M \equiv N \wedge P$, and suppose that $Var(N) \cap Var(P) \neq \emptyset$. Consider the case when $I/N \models N$ and $I/P \not\models P$, so that $I \not\models N \wedge P$. Expanding $[M]_g$, $\textsc{and}(a, b, g)$, and using Lemma 2.4, the following deduction can be constructed.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash g, g^\perp}\mathbf{I} \qquad \dfrac{\dfrac{\vdots}{\vdash \langle I/N \rangle, [N]_a, a} \quad \dfrac{\vdots}{\vdash \langle I/P \rangle, [P]_b, b^\perp}}{\vdash \langle I/N \rangle, \langle I/P \rangle, (a \otimes b^\perp), [N]_a, [P]_b}\otimes}{\vdash \langle I/N \rangle, \langle I/P \rangle, (a \otimes b^\perp \otimes g), [N]_a, [P]_b, g^\perp}\otimes}{\vdash \langle I/N \rangle, \langle I/P \rangle, \textsc{and}(a, b, g), [N]_a, [P]_b, g^\perp}\oplus}{\vdots}}{\dfrac{\vdots}{\vdash \langle I \rangle, \textsc{and}(a, b, g), \textsc{copyall}(Var(N) \cap Var(P)), [N]_a, [P]_b, g^\perp}}{\vdash \langle I \rangle, \textsc{and}(a, b, g) \parr \textsc{copyall}(Var(N) \cap Var(P)) \parr [N]_a \parr [P]_b, g^\perp}\parr$$

Applying the induction hypothesis to I/N, N, and a, and to I/P, P, and b, we can establish that the remaining subgoals of the deduction are provable.

The remaining subcases in the evaluation of $N \wedge P$ are similar, as are the remaining cases in the induction argument. ∎

The next step is to establish the converse of Lemma 2.5. The classical interpretation of the MALL connectives may be used to give a relatively easy proof. In the classical interpretation, truth values, T and F, are assigned to the MALL atoms, $A^\perp$ is read as classical negation, $A \otimes B$ and $A \& B$ are read as classical conjunction, and $A \parr B$ and $A \oplus B$ are read as classical disjunction. A sequent is interpreted as the classical disjunction of the formulas contained in it.

Lemma 2.6 *If $\vdash \Gamma$ is a provable* MALL *sequent, then for any assignment of truth values to the atoms in Γ, there exists a formula A in the sequence Γ such that A is true under the classical interpretation.*

Proof. The proof is by a straightforward induction on cut-free MALL proofs. Clearly, for axioms $\vdash x, x^{\perp}$, one of x or $x^{\perp}$ must evaluate to T in a given truth assignment. In the induction case, suppose that the last step in the proof of Γ is a $\otimes$-rule of the form

$$\frac{\begin{array}{c}\vdots\\ \vdash B, \Gamma_1\end{array} \qquad \begin{array}{c}\vdots\\ \vdash C, \Gamma_2\end{array}}{\vdash (B \otimes C), \Gamma_1, \Gamma_2}\otimes$$

By the induction hypothesis, the sequence B, Γ_1 contains a formula A_1, and the sequence C, Γ_2 contains a formula A_2, and both A_1 and A_2 are true. If A_1 is different from B, then A_1 occurs in the conclusion sequent yielding the required A, and similarly, when A_2 is different from C. Otherwise, the formula $(B \otimes C)$ is $(A_1 \otimes A_2)$ and is hence true under the classical interpretation of $\otimes$ as conjunction. The induction arguments corresponding to the other connectives are similar. ∎

The main intuition behind Lemma 2.7 is that by appropriately assigning truth values to the literals in $\langle I \rangle$ and $[M]_g$, it is possible to mimic the evaluation of the Boolean formula M under I. Due to our use of one-sided sequents and the form of our encoding, there is exactly one truth value falsifying each formula in $\langle I \rangle$ and $[M]_g$. This assignment turns out be the appropriate one, i.e., the value of g under this assignment is T exactly when $I \models M$. For example, if I is $\{X \leftarrow F\}$ and M is $\neg X$, then $\langle I \rangle$ is $x^{\perp}$ and $[M]_g$ is $(x \otimes g) \oplus (x^{\perp} \otimes g^{\perp})$. The only falsifying assignment here is $\{x \leftarrow F, g \leftarrow T\}$.

Lemma 2.7 *Let M be a Boolean formula and I be an assignment for the variables in M. There exists an assignment K of truth values to the atoms in $\langle I \rangle$ and $[M]_g$ such that for every formula A in the sequence $\langle I \rangle, [M]_g$, assignment K falsifies A under the classical interpretation, and $K(g) = T$ iff $I \models M$.*

Proof. The proof is by induction on the construction of $[M]_g$. Note that the induction is parametric in I and g (I and g are universally quantified in the induction hypothesis), so that when $M \equiv (N \wedge P)$, the induction hypothesis on N has I/N replacing I and a replacing g, where a labels the output of N.

Base case: $M \equiv X$. Then $[M]_g = (x^{\perp} \otimes g) \oplus (x \otimes g^{\perp})$. If $I \models X$, then $I(X) = T$ and $\langle I \rangle = x^{\perp}$, and $\langle I \rangle$ is falsified if K assigns T to x. $[M]_g$ is falsified if K assigns T to g, and the second part of the conclusion, $K(g) = T$ also follows. If $I \not\models X$, then $I(X) = F$. Let K assign F to x and F to g to falsify both $\langle I \rangle$ and $[M]_g$. Then $K(g) = F$ as required.

Induction step: Observe first that the formula $\textsc{copy}(x)$ defined as $(x \otimes (x^\perp \parr x^\perp)) \oplus (x^\perp \otimes (x \parr x))$ is classically false.

When $M \equiv \neg N$, the encoding $[M]_g$ is $\textsc{not}(a,g), [N]_a$. By the induction hypothesis, we have an assignment K_1 falsifying $\langle I \rangle, [N]_a$ such that $K_1(a) = T$ iff $I \models N$. Suppose $K_1(a) = T$, and hence $I \models N$. The formula $\textsc{not}(a,g)$ is $(a \otimes g) \oplus (a^\perp \otimes g^\perp)$. Let K be $K_1\{g \leftarrow F\}$. Since g does not occur in $\langle I \rangle$ or $[N]_a$, K agrees with K_1 on $\langle I \rangle, [N]_a$. The assignment K also falsifies $\textsc{not}(a,g)$, thus falsifying $\langle I \rangle, [M]_g$. Note that $K(g) = F$ as required, since $I \not\models M$.

If $K_1(a) = F$, then $I \not\models N$. Letting K be $K_1\{g \leftarrow T\}$ falsifies $\langle I \rangle, [M]_g$.

When $M \equiv (N \wedge P)$, then by the induction hypotheses for N and P, there exists

1. K_1 falsifying $\langle I/N \rangle, [N]_a$ such that $K_1(a) = T$ iff $I/N \models N$, and
2. K_2 falsifying $\langle I/P \rangle, [P]_b$ such that $K_2(a) = T$ iff $I/P \models P$

The encoding $\langle I \rangle$ is a sequence of literals such that no two distinct literals in $\langle I \rangle$ share a common atom. Since $\langle I/N \rangle$ and $\langle I/P \rangle$ are subsets of $\langle I \rangle$, there is no literal x such that x is in $\langle I/N \rangle$ and $x^\perp$ is in $\langle I/P \rangle$. Formulas $[N]_a$ and $[P]_b$ have no atoms in common outside of those in $\langle I \rangle$. Then the union of the assignments $K_1 \cup K_2$, is still an assignment, i.e., it assigns a unique truth value to each *atom* in $\langle I \rangle, [N]_a, [P]_b$.

Suppose that $K_1(a) = T$, and hence $I/N \models N$, and $K_2(b) = F$, so that $I/P \not\models P$. Let K be $(K_1 \cup K_2)\{g \leftarrow F\}$. Note that g does not occur in $\langle I \rangle$, $[N]_a$ or $[P]_b$ so that K agrees with K_1 on $[N]_a$ and with K_2 on $[P]_b$. Each disjunct in $\textsc{and}(a,b,g)$ expanded as

$$\begin{array}{cc} (a \otimes b \otimes g^\perp) & \oplus \\ (a^\perp \otimes b^\perp \otimes g) & \oplus \\ (a \otimes b^\perp \otimes g) & \oplus \\ (a^\perp \otimes b \otimes g) & \end{array}$$

is falsified by K. As already observed, the $\textsc{copy}$ formulas are all classically false, and thus K falsifies $\langle I \rangle, [M]_g$. Since in this case, $I \not\models N \wedge P$, the second part of the conclusion is also satisfied.

The remaining cases are similar. ∎

Lemma 2.8 *If I is an assignment for the variables in a given Boolean formula M, then*

1. *if $\vdash \langle I \rangle, [M]_g, g$ is provable, $I \models M$*
2. *if $\vdash \langle I \rangle, [M]_g, g^\perp$ is provable, $I \not\models M$.*

Proof. By Lemma 2.6, we know that if $\vdash \langle I \rangle, [M]_g, g$ is provable, then no assignment can simultaneously falsify $\langle I \rangle$, $[M]_g$, and g under the classical interpretation.

By Lemma 2.7, we can find an assignment K which falsifies $\langle I \rangle$ and $[M]_g$ such that $K(g) = T$ iff $I \models M$. Since K cannot also falsify g, $K(g) = T$ and hence $I \models M$.

Similarly, when $\vdash \langle I \rangle, [M]_g, g^\perp$ is provable, we can, by Lemmas 2.7 and 2.6, find an assignment K such that $K(g) = F$ and as a consequence, $I \not\models M$. ∎

Lemma 2.9 $\vdash \langle I \rangle, [M]_g, g$ *is provable iff* $I \models M$.

Proof. Follows immediately from Lemmas 2.5 and 2.8. ∎

So far, we have demonstrated the correctness of the encoding of the Boolean matrix of a given quantified Boolean formula. The remainder of the proof deals with the encoding of Boolean quantifiers. The next lemma states the crucial reason why the MALL encoding of quantifiers is faithful to the quantifier orderings. As observed in Section 2.2.2, the goal is to ensure that in any successful proof search, the ith quantifier encoding is reduced after, i.e, above, the reduction of the $(i+1)$st quantifier encoding in any cut-free proof. To achieve this, we need to argue that the key q_i needed to unlock the ith quantifier encoding is only made available when the $(i+1)$st quantifier encoding has been reduced. In order for the ith quantifier encoding, which has the form $q_i^\perp \otimes U_i$, to be reduced before the $(i+1)$st quantifier encoding, a subgoal of the form $\vdash q_i^\perp, \Gamma$ would have to be provable. The only occurrences of q_i are in the subformula U_{i+1} given by $(q_i \parr x_{i+1}) \circ (q_i \parr x_{i+1}^\perp)$, where $\circ$ may be either $\oplus$ or $\&$. If U_{i+1} occurs in Γ, then the only occurrences of q_i in Γ are as immediate arguments to a $\parr$. By exploiting the absence of an unrestricted weakening rule in MALL, it can be shown that in the absence of constants, $\vdash q_i^\perp, \Gamma$ is not provable when all of the occurrences of q_i in Γ appear as immediate arguments to $\parr$. Therefore, regardless of whether U_{i+1} occurs in Γ, the sequent $\vdash q_i^\perp, \Gamma$ would not be provable, thus making it impossible to reduce the ith quantifier encoding below the $(i+1)$st quantifier encoding in a cut-free proof.

Lemma 2.10 *If q is a positive or negative literal and the sequent $\vdash q, \Gamma$ contains no constants, then $\vdash q, \Gamma$ is provable only if either $\Gamma \equiv q^\perp$ or Γ contains at least one occurrence of a subformula either of the form $q^\perp \circ A$, or the form $A \circ q^\perp$, where $\circ$ may be either $\oplus$, $\&$, or $\otimes$.*

Proof. We fix $\circ$ to be either $\oplus$, $\&$, or $\otimes$ for this proof. The proof is by induction on cut-free MALL proofs of $\vdash q, \Gamma$. In the base case, for a cut-free proof of depth 0, the sequent $\vdash q, \Gamma$ must be a MALL axiom, and $\Gamma \equiv q^\perp$ holds.

In the induction step, when in the given cut-free proof of $\vdash q, \Gamma$, the conclusion sequent is derived by an application of either a $\otimes$, $\&$ or a $\oplus$ rule, then at least one premise must be of the form $\vdash q, \Delta$. We know by the induction hypothesis for the proof of $\vdash q, \Delta$, either $\Delta = q^\perp$ or Δ either contains a subformula of the form $q^\perp \circ A$, or the form $A \circ q^\perp$. In either case, Γ contains one of the forms, $q^\perp \circ A$ or $A \circ q^\perp$.

If in the cut-free proof of $\vdash q, \Gamma$, the conclusion sequent is derived by an application of the $\parr$ rule, the premise sequent must be of the form $\vdash q, \Delta$, where Δ is not a single formula, Then by the induction hypothesis on the proof of $\vdash q, \Delta$, the sequence Δ must contain one of the forms, $q^\perp \circ A$ or $A \circ q^\perp$. Since every subformula of Δ is a subformula of Γ as well, Γ must also contain one of the forms $q^\perp \circ A$ or $A \circ q^\perp$. ■

The following lemma demonstrates the correctness of the MALL encoding of Boolean quantifiers. Each Q_i in the statement below is either $\forall$ or $\exists$.

Lemma 2.11 (Main Induction) *Let M be a Boolean formula in the variables $X_1, \ldots, X_n$, then for any m, $0 \leq m \leq n$, and assignment I for $X_{m+1}, \ldots, X_n$, the relation $I \models Q_m X_m \ldots Q_1 X_1 \colon M$ holds iff the sequent $\vdash q_m, \langle I \rangle, [\![Q_m X_m \ldots Q_1 X_1 \colon M]\!]_g, g$ is provable in* MALL.

Proof. The proof is by induction on m between 0 and n. Note that I is universally quantified in the induction hypothesis.

Base case $\Rightarrow$: Here $m = 0$. Then $[\![M]\!]_g \equiv q_0^\perp \otimes [M]_g$, and we can easily construct the following deduction of the required conclusion $\vdash q_0, \langle I_0 \rangle, [\![M]\!]_g, g$.

$$\dfrac{\dfrac{}{\vdash q_0, q_0^\perp}\mathbf{I} \qquad \begin{array}{c}\vdots \\ \vdash \langle I \rangle, [M]_g, g\end{array}}{\vdash \langle I \rangle, q_0, q_0^\perp \otimes [M]_g, g}\otimes$$

The proof of the remaining subgoal $\vdash \langle I \rangle, [G]_g, g$, follows from Lemma 2.5.

Base case $\Leftarrow$: The deduction shown above is the only possible one in a cut-free proof of $\vdash \langle I \rangle, q_0, q_0^\perp \otimes [M]_g, g$ since $q_0^\perp \otimes [G]$ is the only compound formula in the conclusion. So if $\vdash \langle I \rangle, q_0, q_0^\perp \otimes [M]_g, g$ is provable, by Theorem 2.1, it must have a cut-free proof containing a proof of $\vdash \langle I \rangle, [M]_g, g$. By Lemma 2.8, we get $I \models M$.

Induction step $\Rightarrow$: Assume $0 < m \leq n$. Let G abbreviate $Q_{m-1} X_{m-1} \ldots Q_1 X_1 \colon M$. We must prove the lemma for $Q_m X_m G$. If $Q_m \equiv \exists$, then

$$[\![Q_m X_m \colon G]\!]_g = (q_m^\perp \otimes ((x_m \parr q_{m-1}) \oplus (x_m^\perp \parr q_{m-1}))), [\![G]\!]_g$$

If $I \models \exists X_m \colon G$, then either $I, X_m \models G$ or $I, \neg X_m \models G$. In the former case, the following deduction of the required conclusion can be constructed.

$$\dfrac{\dfrac{}{\vdash q_m, q_m^\perp}\mathbf{I} \qquad \dfrac{\begin{array}{c}\vdots \\ \vdash \langle I \rangle, x_m^\perp, q_{m-1}, [\![G]\!]_g, g\end{array}}{\vdash \langle I \rangle, ((x_m \parr q_{m-1}) \oplus (x_m^\perp \parr q_{m-1})), [\![G]\!]_g, g}\oplus}{\vdash \langle I \rangle, q_m, (q_m^\perp \otimes ((x_m \parr q_{m-1}) \oplus (x_m^\perp \parr q_{m-1}))), [\![G]\!]_g, g}\otimes$$

Since $\langle I, X_m\rangle = \langle I\rangle, x_m^\perp$, the induction hypothesis can be applied to show that the remaining subgoal of the above deduction is provable.

When $I, \neg X_m \models G$, the proof construction only differs from the above one on the $\oplus$ rule corresponding to the quantifier encoding.

If $Q_m \equiv \forall$, then

$$[\![\forall X_m\!: G]\!]_g = (q_m^\perp \otimes ((x_m \parr q_{m-1})\&(x_m^\perp \parr q_{m-1}))), [\![G]\!]_g$$

Since $I \models \forall X_m\!: G$, it follows that $I, X \models G$ and $I, \neg X \models G$. The following deduction can be constructed.

$$\dfrac{\dfrac{}{\vdash q_m, q_m^\perp}\mathbf{I} \qquad \dfrac{\begin{array}{c}\vdots\\ \vdash \langle I\rangle, x_m, q_{m-1}, [\![G]\!]_g, g\end{array} \qquad \begin{array}{c}\vdots\\ \vdash \langle I\rangle, x_m^\perp, q_{m-1}, [\![G]\!]_g, g\end{array}}{\vdash \langle I\rangle, ((x_m \parr q_{m-1})\&(x_m^\perp \parr q_{m-1})), [\![G]\!]_g, g}\&}{\vdash \langle I\rangle, q_m, (q_m^\perp \otimes ((x_m \parr q_{m-1})\&(x_m^\perp \parr q_{m-1})), [\![G]\!]_g, g}\otimes$$

Since $\langle I, X_m\rangle$ is $\langle I\rangle, x_m^\perp$ and $\langle I, \neg X_m\rangle$ is $\langle I\rangle, x_m$, the two remaining subgoals in the deduction are provable by the induction hypotheses.

Induction step $\Leftarrow$: This is the critical step in the proof. We are given that $m > 0$ and that the conclusion sequent $\vdash q_m, \langle I\rangle, [\![Q_m X_m \ldots Q_1 X_1\!: M]\!]_g, g$ is provable. Theorem 2.1 can be applied to construct a cut-free proof of $\vdash q_m, \langle I\rangle, [\![Q_m X_m \ldots Q_1 X_1\!: M]\!]_g, g$. We show that this cut-free proof respects the quantifier ordering, i.e., the reduction of the encoding of $Q_m X_m$ occurs below any other step in the proof.

It is easy to see that every formula in the multiset $[\![Q_m X_m \ldots Q_1 X_1\!: M]\!]_g$ is of the form $q_i^\perp \otimes A_i$, for $0 \leq i \leq m$, with $A_0 \equiv [M]_g$, and $A_{j+1} \equiv ((x_{j+1} \parr q_j) \circ (x_{j+1}^\perp \parr q_j))$. The connective written as $\circ$ can be either & or $\oplus$. The formulas $q_i^\perp \otimes A_i$ are the only compound formulas in the conclusion sequent $\vdash q_m, \langle I\rangle, [\![Q_m X_m \ldots Q_1 X_1\!: M]\!]_g, g$. From the MALL rules, it is clear that the only applicable reduction in a cut-free proof search would be an application of the $\otimes$-rule. Hence for some k, we can partition the formulas other than $q_k^\perp \otimes A_k$ in the conclusion sequent into Γ and Δ to get a deduction of the conclusion sequent of the following form.

$$\dfrac{\begin{array}{c}\vdots\\ \vdash \Gamma, q_k^\perp\end{array} \qquad \begin{array}{c}\vdots\\ \vdash A_k, \Delta\end{array}}{\vdash \Gamma, (q_k^\perp \otimes A_k), \Delta}\otimes$$

Suppose for the sake of deriving a contradiction that $k < m$. Recall that there are no constants in the encoding. The formula $q_{k+1}^\perp \otimes A_{k+1}$ must either occur in Γ or Δ, and definitely not in both. Since the only occurrences of q_k are within A_{k+1}, by Lemma 2.10, if $q_{k+1}^\perp \otimes A_{k+1}$ occurs in Δ, then we cannot complete the proof

$\vdash q_k, \Gamma$. Thus we assume $q_{k+1}^{\perp} \otimes A_{k+1}$ occurs in Γ. It is easy to see by inspection of the form of A_{k+1} that the only occurrences of q_k in A_{k+1} have the form $q_k \circ B$ or the form $B \circ q_k$, where $\circ$ is either $\oplus$, &, or $\otimes$. Therefore, again by Lemma 2.10, $\vdash q_k^{\perp}, \Gamma$ is not provable.

Thus it follows that $k \not< m$.

When $k = m$, we can apply Lemma 2.10 to infer that $\Gamma \equiv q_m$, since otherwise, Γ would not contain any occurrences of q_m as immediate arguments to $\otimes$, & or $\oplus$. If $Q_m \equiv \exists$, this yields the deduction

$$\dfrac{\dfrac{}{\vdash q_m, q_m^{\perp}}\mathbf{I} \qquad \begin{array}{c}\vdots\\ \vdash \langle I\rangle, ((x_m \parr q_{m-1}) \oplus (x_m^{\perp} \parr q_{m-1})), [\![Q_{m-1}X_{m-1}\ldots Q_0X_0\colon M]\!]_g, g\end{array}}{\vdash \langle I\rangle, q_m, (q_m^{\perp} \otimes ((x_m \parr q_{m-1}) \oplus (x_m^{\perp} \parr q_{m-1})), [\![Q_{m-1}X_{m-1}\ldots Q_0X_0\colon M]\!]_g, g}\otimes$$

For the same reason as before, the remaining subgoal cannot be reduced by applying the $\otimes$-rule to a formula $q_i^{\perp} \otimes A_i$ since all of the occurrences of q_i remain as immediate arguments to $\parr$. The only possible reduction then is to "unwind" the quantifier encoding for Q_mX_m as in the ($\Leftarrow$) direction of the proof until q_{m-1} is introduced as a sequent formula. If the left $\oplus$ reduction is applied in the given cut-free proof, we have

$$\dfrac{\dfrac{\begin{array}{c}\vdots\\ \vdash \langle I\rangle, x_m, q_{m-1}, [\![Q_{m-1}X_{m-1}\ldots Q_0X_0\colon M]\!]_g, g\end{array}}{\vdash \langle I\rangle, (x_m \parr q_{m-1}), [\![Q_{m-1}X_{m-1}\ldots Q_0X_0\colon M]\!]_g, g}\parr}{\vdash \langle I\rangle, ((x_m \parr q_{m-1}) \oplus (x_m^{\perp} \parr q_{m-1})), [\![Q_{m-1}X_{m-1}\ldots Q_0X_0\colon M]\!]_g, g}\oplus$$

Then by the induction hypothesis applied to the proof of the sequent

$$\vdash \langle I\rangle, x_m, q_{m-1}, [\![Q_{m-1}X_{m-1}\ldots Q_0X_0\colon M]\!]_g, g$$

we get $I, \neg X_m \models Q_{m-1}X_{m-1}\ldots Q_0X_0\colon M$, and hence $I \models \exists X_m Q_{m-1}X_{m-1}\ldots Q_0X_0\colon M$.

The argument is similar when the right $\oplus$ reduction is applied in the given cut-free proof.

The proof when $Q_m \equiv \forall$ is also similar. ∎

When $m = n$ in Lemma 2.11, it follows that a closed QBF G is valid iff $\sigma(G)$ is provable in MALL. Since σ is a log-space encoding of a given QBF, the final result below follows immediately from Theorems 2.11 and 2.2.

Theorem 2.12 MALL *provability is* PSPACE*-complete.*

With two-sided sequents, the intuitionistic fragment of MALL constrains the right-hand side of the sequent to contain at most one formula. A two-sided reformulation of the above proof could be carried out entirely within the intuitionistic fragment of MALL, showing that intuitionistic MALL is also PSPACE-complete.

Chapter 3

Propositional Linear Logic is Undecidable

In the previous chapter, the decision problem for MALL was shown to be PSPACE-complete. We now show that if nonlogical (MALL) axioms are added to MALL, the decision problem becomes recursively unsolvable. We also show that nonlogical MALL axioms may be encoded in full propositional linear logic without nonlogical axioms, and thus we have the result that full propositional linear logic is undecidable.

The proof of undecidability consists of a reduction from the halting problem for a form of counter machine to a decision problem in linear logic. In more detail, we begin by extending propositional linear logic with theories whose (nonlogical) axioms may be used any number of times in a proof. We then describe a form of *and*-branching two-counter machine with an undecidable halting problem and show how to encode these machines in propositional linear logic with theories. Since the axioms of our theories must have a special form, we are able to show the faithfulness of this encoding using a natural form of cut-elimination in the presence of nonlogical axioms. To illustrate the encoding of two-counter machines, we present an example simulation of a simple computation in Section 3.6. On first reading, the reader may wish to jump ahead to that section since it demonstrates the basic mechanism used in the undecidability proof. Also, the crucial cut-standardization step used in this section relies heavily on the cut-elimination procedure for linear logic without nonlogical axioms, first sketched by Girard in [Gir87a]. We give a very explicit proof of cut-elimination for full propositional linear logic in Appendix A, which some readers may find helpful to skim before continuing.

The key to our encoding of an undecidable problem in linear logic is the combination of three powerful mechanisms: resource accumulation, arbitrary reuse, and and-branching. In linear logic, $\vdash A, A$ is very different from $\vdash A$, and this allows us to represent counters in unary. Indefinitely reusable formulas such as $?(B \multimap C)$,

(or axioms of the form $\vdash B^{\perp}, C$) may be used to implement machine instructions. Note that the ? operator is used here to indicate a reusable resource, since we are working with one-sided sequents. If we were to express an axiom as a formula on the left-hand side of the $\vdash$ in a two-sided presentation of linear logic, we would use ! to express the unlimited potential for reuse of instructions.

The operator & may be used to test a conjunction of properties of the simulated machine, such as whether a counter is zero, and if the rest of the computation can continue normally. Together this machinery is enough to encode recursively unsolvable problems in linear sequents.

3.1 Overview

- We define linear logic theories, and prove a cut-standardization theorem for linear logic augmented with theories in Lemma 3.1.
- We prove theories are sound and faithful to a pure linear logic translation in Lemmas 3.3 and 3.2.
- We describe and-branching two counter machines, and note that their halting problem is unsolvable by reduction from standard two counter machines in Lemma 3.4.
- We demonstrate an encoding of our automata into linear logic theories, and prove that the encoding is sound and faithful in Lemmas 3.5 and 3.6.
- We present an example computation, showing the correspondence between the automaton and the (standardized) linear logic proof.
- We combine these lemmas to obtain our main result in Theorem 3.7.

3.2 Linear Logic Augmented With Theories

Essentially, a theory is a set of nonlogical axioms (sequents) that may occur as leaves of a proof tree. The use of theories described here is an extension of earlier work on multiplicative theories [GG89, MOM89].

We define a *positive* literal to be one of the given propositional symbols p_i. A *negative* literal is one of the $p_i^{\perp}$ symbols. An *atomic* formula is any positive or negative literal.

For the theories of interest here, an *axiom* may be any linear logic sequent of the form $\vdash C, p_{i_1}^{\perp}, p_{i_2}^{\perp}, ..., p_{i_n}^{\perp}$, where C is a MALL formula (a linear logic formula without ! or ?) and the remainder of the sequent is made up of negative literals. For example, the sequents $\vdash p_1, p_2^{\perp}$, $\vdash (p_1 \otimes p_2), p_2^{\perp}$, $\vdash (p_1 \parr p_1)$, and $\vdash p_1^{\perp}, p_2^{\perp}$ may all be axioms.

However, $\vdash p_1, p_1$ and $\vdash (p_1 \otimes p_2), p_3$ may not be axioms. We use this restriction on axioms to achieve strict control over the shape of a proof, as described in Lemma 3.1. Some of this control is lost if the definition of theory is generalized, although for some applications the weaker available results would be sufficient.

Any finite set of axioms is a *theory*. We consider only finite theories so that theories may be encoded as formulas of linear logic. For any theory T, we say that a sequent $\vdash \Gamma$ is provable in T exactly when we are able to derive $\vdash \Gamma$ using the standard set of linear logic proof rules, in combination with axioms from T. Thus each axiom of T is treated as a reusable sequent which may occur as a leaf of a proof tree. For notational convenience, in the case that the axiom $\vdash \Gamma$ occurs in the theory T, we will write

$$\frac{}{\vdash \Gamma}\mathbf{T}$$

A *directed* cut is one where at least one premise is an axiom $\vdash C, p_{i_1}^{\perp}, p_{i_2}^{\perp}, ..., p_{i_n}^{\perp}$ in T, and C is the cut-formula in that axiom. We call any axiom premise of a directed cut where the cut-formula in that axiom is not a negative literal a *principal axiom* of that directed cut. By definition, all directed cuts have at least one principal axiom. A cut between two axioms is always directed, and if the cut-formula of such a cut is non-atomic, that cut has two principal axioms. A *directed* or *standardized* proof is a proof with all cuts directed.

When theories are added to linear logic the cut-elimination Theorem A.3 no longer holds, due to the added axioms which may participate directly in cuts. However, we do obtain the following result:

Lemma 3.1 (Cut Standardization) *If there is a proof of* $\vdash \Gamma$ *in theory* T*, then there is a directed proof of* $\vdash \Gamma$ *in theory* T*.*

Proof. We modify the cut-elimination procedure defined in Appendix A in two ways. First we alter the definition of degree to ignore the measure of directed cuts. Formally, we say that if a cut is directed, its degree is zero. Second, we modify the procedure given in Lemma A.1 to handle the extra cases brought about by the presence of axioms. We must allow all the reductions as stated in Appendix A to apply to the case when one of the premises is an axiom, but we need not introduce any truly novel reductions.

We will follow the notation used in Appendix A, where **Cut**$*$ is used to ambiguously refer to the **Cut** rule or the extra rule of inference introduced in the appendix called **Cut**!. Also, we will define all the formulas which appear in an axiom to be principal in that application of the axiom.

In Appendix A most of the reductions are given for some specific derivation versus any possible derivation. For example, all the non-principal cases are stated for any derivation of the "other" hypothesis of **Cut**$*$. Similarly, the Identity and $\top$

rules are stated for any derivation of the "other" hypothesis. We simply now state that even if the other derivation involves an axiom, the reduction still applies.

For example, if the last rule applied in the left hypothesis is $\otimes$, and the last rule in the right hypothesis is an axiom, we apply the following transformation:

$$\dfrac{\dfrac{\begin{array}{c}\vdots\\ \vdash \Sigma, C\end{array} \quad \begin{array}{c}\vdots\\ \vdash B, \Delta, p_i\end{array}}{\vdash \Sigma, (C \otimes B), \Delta, p_i}\otimes \quad \dfrac{}{\vdash \Gamma, p_i^{\perp}}\mathbf{T}}{\vdash \Sigma, (C \otimes B), \Delta, \Gamma}\mathbf{Cut} \quad \Longrightarrow \quad \dfrac{\begin{array}{c}\vdots\\ \vdash \Sigma, C\end{array} \quad \dfrac{\begin{array}{c}\vdots\\ \vdash B, \Delta, p_i\end{array} \quad \dfrac{}{\vdash \Gamma, p_i^{\perp}}\mathbf{T}}{\vdash B, \Delta, \Gamma}\mathbf{Cut}}{\vdash \Sigma, (C \otimes B), \Delta, \Gamma}\otimes$$

This is simply a special case of the reduction given in case A.1.1 in Appendix A.

Also, as a second example, the reduction given for Identity is applicable even to the axiom case:

$$\dfrac{\dfrac{}{\vdash p_i, p_i^{\perp}}\mathbf{I} \quad \dfrac{}{\vdash \Gamma, p_i^{\perp}}\mathbf{T}}{\vdash \Gamma, p_i^{\perp}}\mathbf{Cut} \quad \Longrightarrow \quad \dfrac{}{\vdash \Gamma, p_i^{\perp}}\mathbf{T}$$

Again, this is simply a special case of the reduction given in Appendix A.

As a third and final example of specializations of reductions given in the appendix, the $\top$ rule also applies to axioms:

$$\dfrac{\dfrac{}{\vdash \top, \Sigma, p_i}\top \quad \dfrac{}{\vdash \Gamma, p_i^{\perp}}\mathbf{T}}{\vdash \top, \Sigma, \Gamma}\mathbf{Cut} \quad \Longrightarrow \quad \dfrac{}{\vdash \top, \Sigma, \Gamma}\top$$

This is also simply a special case of the reduction given in the appendix.

Now, some simple analysis is required to show that there are no new cases of principal cuts involving axioms. If the cut in question is already directed, the cut has degree zero, by our modified definition, and thus we are done. Otherwise, by definition of axiom we know that the cut-formula is a negative atomic literal. There are only two rules where an atomic literal may be principal: Identity and $\top$. However, both of these cases are handled by existent reductions (restated above). One should also note that since any cut involving two axioms must be directed, we needn't provide a reduction for that case.

This completes the discussion of the modifications to Lemma A.1 necessary to handle nonlogical axioms. Fortunately, Lemma A.2 and Theorem A.3 then follow without modification (although the definition of degree has changed somewhat).

Therefore, given any proof of a sequent $\vdash \Gamma$ in theory T, we can construct a directed proof of $\vdash \Gamma$ in theory T. ∎

The cut-elimination procedure in Appendix A introduces a new rule of inference called **Cut!**. If we generalized axioms to allow ? and ! formulas in axioms, we would have to generalize the notion of directed proof to include cases involving **Cut!**, and

a post processing step would be required to transform all directed **Cut**!s into a sequence of contractions followed by a single directed **Cut**, or perhaps simply into a sequence of **Cut**s. In any event, our axioms are restricted to MALL formulas so that any cut involving an axiom is always an application of **Cut**, never of **Cut**!.

3.3 Coding Theories in Formulas

In the next few sections, we show that adding nonlogical axioms to MALL increases the difficulty of deciding if a sequent is provable from PSPACE to undecidable. However, we first show that adding nonlogical axioms to full propositional linear logic does not increase its expressive power, or the difficulty of its decision problem. To accomplish this we show how to encode nonlogical axioms in full propositional linear logic, and then prove that the translation is sound and faithful.

We define the translation $[T]$ of a theory T with k axioms into a multiset of pure linear logic formulas by

$$[\{t_1, t_2, \cdots, t_k\}] = ?[t_1], ?[t_2], \cdots, ?[t_k]$$

where $[t_i]$ is defined for each axiom t_i as follows:

$$[\vdash C, p_a^\perp, p_b^\perp, ..., p_z^\perp] \triangleq (C \parr p_a^\perp \parr p_b^\perp \parr \cdots \parr p_z^\perp)^\perp \;=\; (C^\perp \otimes p_a \otimes p_b \otimes \cdots \otimes p_z)$$

Note that the $p_j^\perp$'s are negative literals, and that since linear logic is commutative, we needn't be concerned with the order of formulas in an axiom. Thus each axiom becomes a reusable formula, where the parity of the subformulas of the axiom have been inverted in the formula.

Intuitively, it should be clear that for any theory T, the sequent $\vdash \Gamma$ should be provable in theory T if and only if $\vdash [T], \Gamma$ is provable without nonlogical axioms. However, due to the unusual nature of linear logic, we will present the proof in detail.

Lemma 3.2 (Theory $\Rightarrow$) *For any finite set of axioms T, $\vdash \Gamma$ is provable in theory T only if $\vdash [T], \Gamma$ is provable without nonlogical axioms.*

Proof. Given some proof of $\vdash \Gamma$ in theory T, we have a linear logic proof tree with axioms of T at some leaves. For each leaf of the proof tree of the form $\vdash \Delta$, where $\vdash \Delta$ is some axiom t_i, we replace that leaf with a small proof of $\vdash [T], \Delta$. This proof tree will be constructed from the proof tree for $\vdash [t_i], \Delta$, and then one application of dereliction leaves us with $\vdash ?[t_i], \Gamma$. Since each formula in $[T]$ begins with ?, we may weaken in the remainder of $[T]$, and thus with some number of weakening steps we have $\vdash [T], \Delta$. For example, if there are k axioms, and $\vdash \Delta$ is

the axiom $t_1 = \vdash q_1^\perp, (q_2 \otimes a)$, then we know $[t_1] = q_1 \otimes (q_2^\perp \parr a^\perp)$. We then perform the following transformation:

$$\frac{}{\vdash q_1^\perp, (q_2 \otimes a)}\mathbf{T} \quad \Longrightarrow \quad \begin{array}{c} \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash q_1, q_1^\perp}\mathbf{I} \quad \dfrac{\dfrac{\dfrac{}{\vdash q_2^\perp, q_2}\mathbf{I} \quad \dfrac{}{\vdash a^\perp, a}\mathbf{I}}{\vdash q_2^\perp, (q_2 \otimes a), a^\perp}\otimes}{\vdash (a^\perp \parr q_2^\perp), (q_2 \otimes a)}\parr}{\vdash q_1^\perp, q_1 \otimes (q_2^\perp \parr a^\perp), (q_2 \otimes a)}\otimes}{\vdash q_1^\perp, ?[t_1], (q_2 \otimes a)}?\mathbf{D}}{\vdash ?[t_1], ?[t_2], q_1^\perp, (q_2 \otimes a)}?\mathbf{W}}{\vdots}?\mathbf{W} \\ \hline \vdash ?[t_1], ?[t_2], ...?[t_{k-1}], q_1^\perp, (q_2 \otimes a) \\ \hline \vdash [T], q_1^\perp, (q_2 \otimes a) \end{array}?\mathbf{W}$$

For each leaf sequent which was originally an application of identity, we weaken in all the $?[t_i]$ formulas:

$$\frac{}{\vdash p_i, p_i^\perp}\mathbf{I} \quad \Longrightarrow \quad \begin{array}{c} \dfrac{\dfrac{\dfrac{\dfrac{}{\vdash p_i, p_i^\perp}\mathbf{I}}{\vdash ?[t_1], p_i, p_i^\perp}?\mathbf{W}}{\vdash ?[t_1], ?[t_2], p_i, p_i^\perp}?\mathbf{W}}{\vdots}?\mathbf{W} \\ \hline \vdash ?[t_1], ?[t_2], ...?[t_{k-1}], p_i, p_i^\perp \\ \hline \vdash [T], p_i, p_i^\perp \end{array}?\mathbf{W}$$

We then continue by adding $[T]$ to every sequent in the entire proof tree. At every application of $\otimes$ and **Cut**, we extend the proof tree with an extra copy of the conclusion sequent of the binary rule, to which we add an extra copy of $[T]$. Then we extend the proof further, adding one contraction step for each $?[t_i]$ between that sequent and the original conclusion of that binary rule.

$$\frac{\begin{array}{c}\vdots \\ \vdash \Sigma, A\end{array} \qquad \begin{array}{c}\vdots \\ \vdash B, \Delta\end{array}}{\vdash \Sigma, (A \otimes B), \Delta}\otimes \quad \Longrightarrow \quad \begin{array}{c} \dfrac{\begin{array}{c}\vdots \\ \vdash [T], \Sigma, A\end{array} \qquad \begin{array}{c}\vdots \\ \vdash [T], B, \Delta\end{array}}{\vdots}\otimes \\ \hline \vdots \\ \hline \vdash [T], \Sigma, (A \otimes B), \Delta \end{array}?\mathbf{C}$$

Thus we have given a construction which builds a proof of $\vdash [T], \Gamma$ without any nonlogical axioms from a given proof of $\vdash \Gamma$ using axioms from T. ∎

Lemma 3.3 (Theory $\Leftarrow$) *For any finite set of axioms T, $\vdash \Gamma$ is provable in theory T if $\vdash [T], \Gamma$ is provable without nonlogical axioms.*

Proof. For each axiom $t_i \equiv \vdash C, p_a^\perp, p_b^\perp, ..., p_z^\perp$, we may prove $!([t_i]^\perp) \equiv !(C \parr p_a^\perp \parr p_b^\perp \parr \cdots \parr p_z^\perp)$ by several applications of $\parr$ and one application of **!S**, as follows.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash C, p_a^\perp, p_b^\perp, \cdots, p_z^\perp}\mathbf{T}}{\vdash (C \parr p_a^\perp), p_b^\perp, \cdots, p_z^\perp}\parr}{\vdash (C \parr p_a^\perp \parr p_b^\perp), \cdots, p_z^\perp}\parr}{\vdots}\parr}{\vdash (C \parr p_a^\perp \parr p_b^\perp \parr \cdots \parr p_z^\perp)}\parr}{\vdash !(C \parr p_a^\perp \parr p_b^\perp \parr \cdots \parr p_z^\perp)}\mathbf{!S}$$

By cutting this proof against a given proof of $\vdash [T], \Gamma$, we obtain a proof of $\vdash [T - \{t_i\}], \Gamma$, where $T - \{t_i\}$ is the multiset difference of T and $\{t_i\}$.

$$\dfrac{\vdots \quad \dfrac{\vdots \quad \dfrac{\dfrac{\vdots \quad \vdots}{\vdash !([t_k]^\perp) \quad \vdash [T], \Gamma}\mathbf{Cut}}{\vdash [\{t_1, \cdots, t_{k-1}\}]\Gamma} \quad \dfrac{\vdots}{\vdash ?([t_1]), ?([t_2]), \Gamma}}{\vdots \quad \vdash !([t_2]^\perp) \quad \vdash ?([t_1]), \Gamma}\mathbf{Cut}}{\vdots \quad \vdash !([t_1]^\perp) \quad \vdash \Gamma}\mathbf{Cut}$$

Thus by induction on the number of axioms, we can derive $\vdash \Gamma$ in theory T. ∎

We have just shown how a decision problem for MALL with the addition of nonlogical axioms may be encoded in full propositional linear logic without nonlogical axioms. Thus the upcoming proof of undecidability of MALL with nonlogical axioms will yield undecidability for full propositional linear logic.

3.4 And-Branching Two Counter Machines Without Zero-Test

In this section we describe nondeterministic two counter machines with *and*-branching but without a zero-test instruction. We show that these machines have a recursively unsolvable halting problem, and then we will show how the halting problem for these machines may be encoded as a decision problem in MALL, with nonlogical axioms corresponding to the machine instructions.

The machines described here are similar to standard two counter machines except for the lack of an explicit zero test transition, and the addition of "fork" transitions. Intuitively, Q_i *Fork* Q_j, Q_k is an instruction which allows a machine in state Q_i to continue computation from both states Q_j and Q_k, each computation continuing with the current counter values. For brevity in the following proofs, we emphasize *two* counter machines. However, there is no intrinsic reason to restrict the machines to two counters. All of our arguments and results generalize easily to N counters, for $N \geq 2$. Formally, an *And-Branching Two Counter Machine Without Zero-Test*, or ACM for short, is given by a finite set Q of states, a finite set δ of transitions, and distinguished initial and final states, Q_I and Q_F, as described below.

An *instantaneous description*, or *ID*, of an ACM M is a finite list of ordered triples $\langle Q_i, A, B\rangle$, where $Q_i \in Q$, and A and B are natural numbers, each corresponding to a *counter* of the machine. Intuitively, a list of triples represents a set of machine configurations. One may think of an ACM state as some sort of parallel computation which terminates successfully only if all its concurrent computation fragments terminate successfully.

We define the *accepting* triple as $\langle Q_F, 0, 0\rangle$. We also define an *accepting* ID as any ID where every element of the ID is the accepting triple. That is, every and-branch of the computation has reached an accepting triple. We say that an ACM M *accepts from* an ID s if and only if there is some computation from s to an accepting ID. It is essential for our encoding in linear logic that both counters be zero in all elements of an accepting ID.

The set δ may contain transitions of the following form:

(Q_i **Increment** A Q_j) taking
$\langle Q_i, A, B\rangle$ to $\langle Q_j, A+1, B\rangle$
(Q_i **Increment** B Q_j) taking
$\langle Q_i, A, B\rangle$ to $\langle Q_j, A, B+1\rangle$
(Q_i **Decrement** A Q_j) taking
$\langle Q_i, A+1, B\rangle$ to $\langle Q_j, A, B\rangle$
(Q_i **Decrement** B Q_j) taking
$\langle Q_i, A, B+1\rangle$ to $\langle Q_j, A, B\rangle$
(Q_i **Fork** Q_j, Q_k) taking
$\langle Q_i, A, B\rangle$ to $(\langle Q_j, A, B\rangle,\ \ \langle Q_k, A, B\rangle)$

where Q_i, Q_j, and Q_k are states in Q. The **Decrement** instructions only apply if the appropriate counter is not zero, while the **Increment** and **Fork** instructions are always enabled from the proper state.

For example, the single transition Q_i **Increment** A Q_j takes an ACM from ID: $\{\cdots, \langle Q_i, A, B\rangle, \cdots\}$ to ID: $\{\cdots, \langle Q_j, A+1, B\rangle, \cdots\}$

3.4.1 Two Counter Machines

Standard two counter machines have a finite set of states, Q, a finite set of transitions, δ, a distinguished initial state Q_I, and a set of final states F [Min61, HU79]. An instantaneous description of the state of a two counter machine is given by a triple $\langle Q_i, A, B\rangle$, which consists of the current state, and the values of two counters, A and B. The transitions in δ are of four kinds:

(Q_i **Increment** A Q_j)	taking	$\langle Q_i, A, B\rangle$ to $\langle Q_j, A+1, B\rangle$
(Q_i **Increment** B Q_j)	taking	$\langle Q_i, A, B\rangle$ to $\langle Q_j, A, B+1\rangle$
(Q_i **Decrement** A Q_j)	taking	$\langle Q_i, A+1, B\rangle$ to $\langle Q_j, A+1, B\rangle$
(Q_i **Decrement** B Q_j)	taking	$\langle Q_i, A, B+1\rangle$ to $\langle Q_j, A, B+1\rangle$
(Q_i **Zero-Test** A Q_j)	taking	$\langle Q_i, 0, B\rangle$ to $\langle Q_j, 0, B\rangle$
(Q_i **Zero-Test** B Q_j)	taking	$\langle Q_i, A, 0\rangle$ to $\langle Q_j, A, 0\rangle$

A two counter machine accepts if it is able to reach any one of the final states in the set F with both counters at zero. It is important that these machines have a **Zero-Test** instruction since the halting problem becomes decidable otherwise, by obvious reduction to the word problem in commutative semi-Thue systems, which is decidable [MM82]. Since **Zero-Test** is the most difficult to encode in linear logic, we concentrate on a machine with and-branching, which provides a basic mechanism powerful enough to simulate **Zero-Test**, but which is more easily simulated in linear logic.

Using two counter machines, we show that ACM's have an undecidable halting problem.

Lemma 3.4 *It is undecidable whether an and-branching two counter machine without zero-test accepts from ID* $\{\langle Q_I, 0, 0\rangle\}$. *This remains so if the transition relation of the machine is restricted so that there are no outgoing transitions from the final state.*

Proof. Since ACM's may simulate zero-test with and-branching, ACM's are sufficiently powerful to simulate two counter machines, for which the halting problem is known to be recursively unsolvable [Min61, Lam61]. We will give a construction from standard two counter machines to ACMs, and argue that the construction is sound and faithful. This construction and the proof of its soundness is routine, and its steps should be familiar to anyone versed in automata theory. In our simulation of the test for zero instruction of two counter machines, we make essential use of the fact that all branches of computation terminate with both counters set to zero.

Given a nondeterministic two counter machine M we first construct an equivalent two counter machine M' with a unique final state Q_F which has no outgoing transitions. One simply adds two new states, Q_D and Q_F to M', and for each

$Q_f \in F$ of M, one adds the instructions (Q_f **Increment** A Q_D) and (Q_D **Decrement** A Q_F). Note that one may simply look at these new transitions as a single nondeterministic step from each old final state to the new (unique) final state, which has no outgoing transitions. However, since there is no general "silent" move, we make the transition in two steps.

We claim without proof that M and M' accept the same set of input values, and are therefore equivalent machines.

From a nondeterministic two counter machine M' with unique final state without outgoing transitions, we construct an ACM M'' as follows. The ACM M'' will have the same set of states, and same initial and final states as M'. The transition function of M'' is built by first taking all the **Increment** and **Decrement** instructions from the transition function of M'. We then add two new states to M'', Z_A and Z_B, which are used to test for zero in each of the two counters. For Z_A, we add two instructions, (Z_A **Decrement** B Z_A), and (Z_A **Fork** Q_F, Q_F), to the transition function of M''. Similarly for Z_B, we add (Z_B **Decrement** A Z_B), and (Z_B **Fork** Q_F, Q_F). Then for each **Zero-Test** instruction of M' of the form

$$(Q_i \textbf{ Zero-Test } \mathrm{A}\ Q_j)$$

we add one instruction to M'':

$$(Q_i \textbf{ Fork } Q_j,\ Z_A).$$

An important feature of M'' is that once a zero testing or final state is entered, no configuration of that branch of computation may ever leave that set of states. More specifically, where M' would test for zero, M'' will fork into two "parallel" computations. One continues in the "same" state as M' would have if the **Zero-Test** had succeeded, and the other branch "verifies" that the counter is indeed zero. While the second branch may change the value of one of the counters (the counter which is not being tested), this cannot affect the values of the counters in the "main" branch of computation. Further, the zero-testing branch of computation never enters any states other than zero-test states or the final state. This holds because there are no outgoing transitions from the final state whatsoever, and the only transitions from the two zero testing states either loop back to that state or move directly to the final state. Also note that any branch of an ACM M'' computation which arrives at the state Z_A may be part of a terminating computation if and only if the counter A is zero when the machine reaches that state. This can be seen by observing that once arriving in Z_A, there is no possibility of modifications to the counter A. The **Decrement** B transition from Z_A to itself allows M'' to loop and decrement the counter B arbitrarily. In particular it is possible for B to be decremented to the value 0. Since Q_F has no outgoing transitions, the **Fork** instruction which moves from Z_A to Q_F and Q_F allows this branch of computation to terminate correctly

if and only if both counters are zero when it is executed. Since we are considering nondeterministic ACMs, it is possible for a branch of computation which reaches Z_A to terminate if and only if the A counter is zero when it reaches Z_A. Similarly, any branch of computation reaching Z_B reaches an accepting ID if and only if the B counter is zero.

We claim that there is a halting computation for the given two counter machine M' if and only if there is one for the constructed ACM M''. This is proven by two simulations.

The and-branching machine M'' may mimic the original two counter machine in the performance of any instruction, by following any **Increment** of M' with the corresponding **Increment** instruction, and a **Decrement** with the corresponding **Decrement**. When M' executes a **Zero-Test** A instruction, M'' forks off an *and* branch which verifies that the counter A is in fact zero, and the other branch continues to follow the computation of M'.

For the converse simulation, there is always at most one and-branch of any M'' computation which corresponds to a nonfinal, non-zero-testing state in the original machine. There may be many *and* branches of the computation which are in states Z_A, Z_B, and Q_F, but at most one *and* branch is in any other state. Thus, M' may mimic M'' by following the branch of ACM computation which does not enter Z_A, Z_B, or Q_F until the final step of computation, when it enters Q_F. For every **Increment** and **Decrement** instruction in the accepting computation of M'', M' may performs the corresponding instruction. Every **Fork** instruction executed by M'' from a nonfinal, non-zero-testing state corresponds to a **Zero-Test** instruction in M', and by the above observation, if M'' forks into state Z_A, then M'' accepts only if the counter A is zero (and similarly for Z_B and the counter B). Since we are assuming an accepting M'' computation, we know that M' may execute the corresponding **Zero-Test** instruction successfully. ∎

3.5 From Machines to Logic

We give a translation from ACMs to linear logic with theories and show that our sequent translation of a machine in a particular state is provable in linear logic if and only if the ACM halts from that state. In fact, our translation uses only MALL formulas and theories, thus with the use of our earlier encoding, Lemma 3.2 and Lemma 3.3, we will have our result for propositional linear logic without nonlogical axioms. Since an instantaneous description of an ACM is given by a list of triples, it is somewhat delicate to state the induction we will use to prove soundness.

We have already seen how the linear connective & may be used to achieve and-branching in the proof of PSPACE-completeness of MALL. We now make use of that, along with some other machinery, to simulate ACM computations.

Given an ACM $M = \langle Q, \delta, Q_I, Q_F \rangle$ we first define a set of propositions:

$$\{q_i | Q_i \in Q\} \bigcup \{q_i^\perp | Q_i \in Q\} \bigcup \{a, a^\perp, b, b^\perp\}$$

We then define the linear logic theory corresponding to the transition relation δ as the set of axioms determined as follows:

$$\begin{array}{rcl} Q_i \textbf{ Increment } \text{A } Q_j & \mapsto & \vdash q_i^\perp, (q_j \otimes a) \\ Q_i \textbf{ Increment } \text{B } Q_j & \mapsto & \vdash q_i^\perp, (q_j \otimes b) \\ Q_i \textbf{ Decrement } \text{A } Q_j & \mapsto & \vdash q_i^\perp, a^\perp, q_j \\ Q_i \textbf{ Decrement } \text{B } Q_j & \mapsto & \vdash q_i^\perp, b^\perp, q_j \\ Q_i \textbf{ Fork } Q_j, Q_k & \mapsto & \vdash q_i^\perp, (q_j \oplus q_k) \end{array}$$

Using linear implication, the (Q_i **Increment** B Q_j) transition may be viewed as $\vdash q_i \multimap (q_j \otimes b)$, *i.e.*, from state Q_i, move to state Q_j and add one to counter B. The other axioms in this translation may also be viewed in this way.

We will write C^n to indicate a sequence of n C's, separated by commas, as follows:

$$C^n \triangleq \overbrace{C, C, \cdots, C}^{n}$$

Since $p^\perp$ is an atomic symbol, the notation ${p^\perp}^3$ will be used for $(p^\perp)^3$, which is simply $p^\perp, p^\perp, p^\perp$.

Given a triple $\langle Q_i, x, y \rangle$ of an ACM, we define the translation $\theta(\langle Q_i, x, y \rangle)$ by:

$$\theta(\langle Q_i, x, y \rangle) \triangleq \vdash q_i^\perp, {a^\perp}^x, {b^\perp}^y, q_F$$

Thus all sequents which correspond to ACM triples have exactly one positive literal, q_F, some number of $a^\perp$s, and $b^\perp$s, the multiplicity of which correspond to the values of the two counters of the ACM in unary, and exactly one other negative literal, which corresponds to the state of the ACM.

The translation of an ACM ID is simply the set of translations of the elements of the ID:

$$\theta(\{E_1, E_2, \cdots, E_m\}) = \{\theta(E_1), \theta(E_2), \cdots, \theta(E_m)\}$$

We claim that an ACM M accepts from ID s if and only if every element of $\theta(s)$ is provable in the theory corresponding to the transition function of the machine. We prove each half of this equivalence in separate lemmas.

Lemma 3.5 (Machine $\Rightarrow$) *An and-branching counter machine M accepts from ID s only if every sequent in $\theta(s)$ is provable in the theory derived from M.*

Proof. Given a halting computation of an ACM machine M from s we claim we can build a proof of every sequent in $\theta(s)$ in the theory derived from M.

M accepts from s only if there is some finite sequence of transitions from this ID to an accepting ID. We proceed by induction on the length of that sequence of transitions.

If there are no transitions in the sequence, then by the definition of accepting ID, s consists entirely of $\langle Q_F, 0, 0\rangle$. We must show that the sequent

$$\vdash q_F^{\perp}, a^{\perp 0}, b^{\perp 0}, q_F$$

is provable in linear logic. This is immediate: we have 0 A's and 0 B's, that is, none at all. Thus by one application of identity per sequent $\vdash q_F^{\perp}, q_F$, we have our proof.

If there is at least one transition in the sequence, we have to show that $\theta(s)$ is provable. Since M accepts from ID $\{\cdots\langle Q_i, A, B\rangle\cdots\}$, and there is at least one transition in the sequence, we know that there is some transition in M such that $ID \rightarrow ID'$, and M accepts from ID'. We assume by induction that there is a linear logic proof which corresponds to the accepting computation for ID'.

We now perform case analysis on the type of transition. There are five different types of instructions: **Increment** A or B, **Decrement** A or B, and **Fork**. Since the two increment and two decrement instructions are nearly identical, we will concentrate only on the cases concerning the counter A.

Q_i **Increment A** Q_j**:** In this case, the first step in the halting computation has the form

$$\{\cdots\langle Q_i, A, B\rangle\cdots\} \rightarrow \{\cdots\langle Q_j, A+1, B\rangle\cdots\}$$

We assume by induction that we have a proof of $\theta(\langle Q_j, A{+}1, B\rangle) = \vdash q_j^{\perp}, a^{\perp A+1}, b^{\perp B}, q_F$. We extend this proof into a proof of $\theta(\langle Q_i, A, B\rangle) = \vdash q_i^{\perp}, a^{\perp A}, b^{\perp B}, q_F$ by adding a cut with an axiom, as follows.

$$\dfrac{\dfrac{}{\vdash q_i^{\perp}, (q_j \otimes a)}\mathbf{T} \qquad \dfrac{\begin{array}{c}\vdots\\ \vdots\\ \vdash q_j^{\perp}, a^{\perp A+1}, b^{\perp B}, q_F\end{array}}{\vdash (q_j^{\perp} \parr a^{\perp}), a^{\perp A}, b^{\perp B}, q_F}\parr}{\vdash q_i^{\perp}, a^{\perp A}, b^{\perp B}, q_F}\mathbf{Cut}$$

Note that the axiom $\vdash q_i^{\perp}, (q_j \otimes a)$ is precisely the translation of the transition taken by the machine, and therefore is an axiom of the theory.

Q_i **Increment B** Q_j**:** Analogous to above.

Q_i **Decrement A** Q_j**:** Since the A counter of the machine must be positive for this instruction to apply, we know that the halting computation begins with the transition

$$\{\cdots\langle Q_i, A+1, B\rangle\cdots\} \rightarrow \{\cdots\langle Q_j, A, B\rangle\cdots\}$$

We assume by induction that we have a proof of $\vdash q_j^\perp, a^{\perp^A}, b^{\perp^B}, q_F$. As in the **Increment** A case, we extend this to a proof of $\vdash q_i^\perp, a^{\perp^{A+1}}, b^{\perp^B}, q_F$ by adding a cut with the axiom corresponding to the transition taken by the machine.

$$\dfrac{\dfrac{}{\vdash q_i^\perp, a^\perp, q_j}\mathbf{T} \qquad \begin{array}{c}\vdots\\ \vdash q_j^\perp, a^{\perp^A}, b^{\perp^B}, q_F\end{array}}{\vdash q_i^\perp, a^{\perp^{A+1}}, b^{\perp^B}, q_F}\mathbf{Cut}$$

Q_i **Decrement B** Q_j**:** Analogous to above.

Q_i **Fork** Q_j, Q_k**:** Here, the halting computation begins with the step

$$\{\cdots\langle Q_i, A, B\rangle\cdots\} \rightarrow \{\cdots\langle Q_j, A, B\rangle, \langle Q_k, A, B\rangle\cdots\}$$

We assume by induction that we have a proof of $\vdash q_j^\perp, a^{\perp^A}, b^{\perp^B}, q_F$, and of $\vdash q_k^\perp, a^{\perp^A}, b^{\perp^B}, q_F$, and we extend those proofs into a proof of $\vdash q_i^\perp, a^{\perp^A}, b^{\perp^B}, q_F$.

$$\dfrac{\dfrac{}{\vdash q_i^\perp, (q_j \oplus q_k)}\mathbf{T} \qquad \dfrac{\begin{array}{c}\vdots\\ \vdash q_j^\perp, a^{\perp^A}, b^{\perp^B}, q_F\end{array} \qquad \begin{array}{c}\vdots\\ \vdash q_k^\perp, a^{\perp^A}, b^{\perp^B}, q_F\end{array}}{\vdash (q_j^\perp \& q_k^\perp), a^{\perp^A}, b^{\perp^B}, q_F}\&}{\vdash q_i^\perp, a^{\perp^A}, b^{\perp^B}, q_F}\mathbf{Cut}$$

Here $\vdash q_i^\perp, (q_j \oplus q_k)$ is the axiom which corresponds to the fork instruction. ∎

Lemma 3.6 (Machine $\Leftarrow$) *An and-branching counter machine M accepts from ID s if every sequent in the set $\theta(s)$ is provable in the theory derived from M.*

Proof.

Given a set of proofs of the elements of $\theta(s)$ in the theory derived from M, we claim that a halting computation of the ACM M from state s can be extracted from those proofs. We achieve this with the aid of the cut standardization Lemma 3.1, which in this case leaves cuts in the proof only where they correspond to applications of ACM instructions. We may thus simply read the description of the computation from the standardized proof.

By Lemma 3.1, it suffices to consider standardized proofs. We show that a set of standardized proofs of $\theta(s)$ may be mimicked by the ACM M to produce an accepting computation from state s.

This proof is by induction on the sum of the sizes (number of proof rules applied) of standardized proofs. Since an ACM state is given by a finite set of triples, and all proofs are finite, we know that this measure is well founded. We assume that any smaller set of proofs which all end in conclusions which correspond to a triple $\langle Q_i, A, B\rangle$ can be simulated by machine M.

We consider the proof of a single element of $\theta(s)$ at a time.

If $s = \{\cdots\langle Q_i, x, y\rangle\cdots\}$, then $\theta(s) \;=\; \{\cdots\vdash q_i^{\perp}, a^{\perp^x}, b^{\perp^y}, q_F \;\cdots\}$.

We assume that we are given a proof of each element of the set $\theta(s)$, and we analyze one of the proofs, all of which end in a conclusion corresponding to a machine triple $\langle Q_i, x, y\rangle$.

$$\begin{array}{c}\vdots\\ \vdash q_i^{\perp}, a^{\perp^x}, b^{\perp^y}, q_F\end{array}$$

Since this sequent is simply a list of atomic propositions, the only linear logic rules which can apply to any such sequent are identity, some axiom, and cut.

Identity is only applicable when both x and y are zero, and $q_i = q_F$. In this case, $\vdash q_F^{\perp}, q_F$ already corresponds to the accepting triple $\langle Q_F, 0, 0\rangle$.

The only axioms which are identical to a sequent in $\theta(s)$ are those which correspond to some δ which is a decrement instruction that ends in q_F. In this case, since each decrement axiom in $[\delta]$ contains exactly one occurrence of $a^{\perp}$ or $b^{\perp}$, $x = 1$ and $y = 0$, or $x = 0$ and $y = 1$. In either case, the ACM machine M need only perform the decrement instruction δ, and this branch of computation reaches an accepting triple.

The final possibility is cut, and by our standardization procedure, we know that one hypothesis of that cut is an axiom from the theory derived from M, and furthermore that the cut-formula in that axiom is not a negative literal.

Since there are only five types of instructions in an ACM; **Increment** A or B, **Decrement** A or B, and **Fork**, there are only five different types of axioms in a theory derived from any ACM M. We now perform case analysis on the type of axiom that was last applied in a proof.

$\vdash q_i^{\perp}, (q_j \otimes a)$**:** If the last axiom applied is of the form $\vdash q_i^{\perp}, (q_j \otimes a)$, then it corresponds to an **Increment** A instruction, and by standardization, we know the cut-formula must be $(q_j \otimes a)$ in the axiom, and that the proof must look like

$$\dfrac{\dfrac{}{\vdash q_i^{\perp}, (q_j \otimes a)}\mathbf{T} \qquad \begin{array}{c}\vdots\\ \vdash (q_j^{\perp} \parr a^{\perp}), a^{\perp^x}, b^{\perp^y}, q_F\end{array}}{\vdash q_i^{\perp}, a^{\perp^x}, b^{\perp^y}, q_F}\mathbf{Cut}$$

Since each other linear logic rule besides $\parr$, cut, identity, or axiom introduces some symbol which does not occur in $\vdash (q_j^\perp \parr a^\perp), a^{\perp x}, b^{\perp y}, q_F$, the derivation of this sequent must end in one of these rules. Furthermore, there are two formulas in this sequent which are not negative literals, so this sequent is not derivable using only an axiom. Identity could not lead to this sequent, since the sequent contains a non-atomic formula. By our standardization procedure, we know that each cut must involve an axiom from the theory, and the cut-formula in the axiom is not a negative literal. Inspecting the various types of axioms in the theory derived from M, we see that all axioms contain one top level negative atomic formula $q_i^\perp$ for some i. Since $q_i^\perp$ cannot be a directed cut-formula in a principal axiom, it must appear in the conclusion of that application of cut. However, there is no such top level q_i in the sequent in question. Thus this sequent may only be derived by the application of the $\parr$ rule. Therefore, we know the derivation must have the form:

$$\dfrac{\dfrac{}{\vdash q_i^\perp, (q_j \otimes a)}\mathbf{T} \quad \dfrac{\begin{array}{c}\vdots \\ \vdash q_j^\perp, a^{\perp x+1}, b^{\perp y}, q_F\end{array}}{\vdash (q_j^\perp \parr a^\perp), a^{\perp x}, b^{\perp y}, q_F}\parr}{\vdash q_i^\perp, a^{\perp x}, b^{\perp y}, q_F}\mathbf{Cut}$$

We know that the proof of $\vdash q_j^\perp, a^{\perp x+1}, b^{\perp y}, q_F$ may be simulated by the ACM by induction, since it is the sequent $\theta(\langle Q_j, x+1, y\rangle)$, which corresponds to the triple $\langle Q_j, x+1, y\rangle$, and has a proof in linear logic of smaller size.

Therefore the machine M may emulate this proof by performing the ACM instruction corresponding to the axiom used (in this case an **Increment** A instruction), and then continuing as dictated by the inductive case.

$\vdash q_i^\perp, (q_j \otimes a)$**:** Analogous arguments apply.

$\vdash q_i^\perp, a^\perp, q_j$**:** If the last axiom applied is $\vdash q_i^\perp, a^\perp, q_j$, which corresponds to a **Decrement** A instruction, then by standardization, we know the cut-formula must be q_j in the axiom, and that the proof must be of the form

$$\dfrac{\dfrac{}{\vdash q_i^\perp, a^\perp, q_j}\mathbf{T} \quad \begin{array}{c}\vdots \\ \vdash q_j^\perp, a^{\perp x}, b^{\perp y}, q_F\end{array}}{\vdash q_i^\perp, a^{\perp x+1}, b^{\perp y}, q_F}\mathbf{Cut}$$

By induction, the proof of $\vdash q_j^\perp, a^{\perp x}, b^{\perp y}, q_F$ can be simulated, since it is the sequent $\theta(\langle Q_j, x, y\rangle)$, which corresponds to the triple $\langle Q_j, x, y\rangle$, and has a shorter proof in linear logic.

Therefore the machine M may emulate this proof by performing the ACM instruction corresponding to the axiom used (in this case a **Decrement** A instruction), and then continuing as dictated by the inductive case.

$\vdash q_i^\perp, a^\perp, q_j$**:** Analogous arguments apply.

$\vdash q_i^\perp, (q_j \oplus q_k)$**:** If the last axiom applied is $\vdash q_i^\perp, (q_j \oplus q_k)$, which corresponds to a **Fork** instruction, then by standardization, we know the cut-formula must be $(q_j \oplus q_k)$ in the axiom, and that the proof must look like

$$\dfrac{\dfrac{}{\vdash q_i^\perp, (q_j \oplus q_k)}\mathbf{T} \qquad \begin{array}{c}\vdots\\ \vdash (q_j^\perp \& q_k^\perp), a^{\perp x}, b^{\perp y}, q_F\end{array}}{\vdash q_i^\perp, a^{\perp x}, b^{\perp y}, q_F}\mathbf{Cut}$$

Since each other linear logic rule besides &, cut, identity, or axiom introduces some symbol which does not occur in $\vdash (q_j^\perp \& q_k^\perp), a^{\perp x}, b^{\perp y}, q_F$, the derivation of this sequent must end in one of these rules. Furthermore, there are two formulas in the sequent which are not negative literals, so this sequent is not derivable using only an axiom. Identity could not lead to this sequent, since the sequent contains a non-atomic formula. By our standardization procedure, we know that each cut must involve an axiom from the theory, and the cut-formula in the axiom is not a negative literal. Inspecting the various types of axioms in the theory derived from M, we see that all axioms contain one top level negative atomic formula $q_i^\perp$ for some i. Since $q_i^\perp$ cannot be the cut-formula in a principal axiom of a directed cut, it must appear in the conclusion of that application of cut. However, there is no such top level q_i in the sequent in question. Thus this sequent may only be derived by the application of the & rule. Thus we know the derivation to be of the form:

$$\dfrac{\dfrac{}{\vdash q_i^\perp, (q_j \oplus q_k)}\mathbf{T} \qquad \dfrac{\begin{array}{c}\vdots\\ \vdash q_j^\perp, a^{\perp x}, b^{\perp y}, q_F\end{array} \qquad \begin{array}{c}\vdots\\ \vdash q_k^\perp, a^{\perp x}, b^{\perp y}, q_F\end{array}}{\vdash (q_j^\perp \& q_k^\perp), a^{\perp x}, b^{\perp y}, q_F}\&}{\vdash q_i^\perp, a^{\perp x}, b^{\perp y}, q_F}\mathbf{Cut}$$

The proofs of $\vdash q_j^\perp, a^{\perp x}, b^{\perp y}, q_F$ and $\vdash q_k^\perp, a^{\perp x}, b^{\perp y}, q_F$ can be simulated on the machine by induction, since one is a sequent which corresponds to the triple $\langle Q_j, x, y\rangle$, the other corresponds to $\langle Q_k, x, y\rangle$, and each has a proof in linear logic of smaller size.

Therefore the machine M may emulate this proof by performing the ACM instruction corresponding to the axiom used (in this case a **Fork** instruction), and then continuing as dictated by the two inductive cases. ∎

From Lemmas 3.4, 3.2, 3.3, 3.5, and 3.6 of this section, we easily obtain our main result:

Theorem 3.7 *The provability problem for propositional linear logic is recursively unsolvable.*

As mentioned earlier, linear logic, like classical logic, has an intuitionistic fragment. Briefly, the intuitionistic fragment is restricted so that there is only one positive formula in any sequent. In fact, the entire construction above was carried out in intuitionistic linear logic, and thus the undecidability result also holds for this logic.

In any theory derived from an ACM M, there is only one positive formula in any theory axiom. Also, throughout a directed proof of $\theta(s)$ in such a theory, the only positive atom which appears outside a theory axiom is q_F. Thus any directed proof of $\theta(s)$ in a theory derived from M is in the intuitionistic fragment of linear logic, and along with a conservativity result not proven here, we have the following:

Corollary 3.8 *The provability problem for propositional intuitionistic linear logic is recursively unsolvable.*

In the proof of this corollary we make use of the conservativity property of full linear logic over the intuitionistic fragment for any sequents occuring in a directed proof of a translation of an ACM machine configuration. This conservativity is a weaker property than full conservativity since sequents in such a directed proof have a special form. In particular, they have no constants, and the right hand side is always a single formula.

3.6 Example Computation

This section is intended to give an overview of the mechanisms we have defined above, and lend some insight into our undecidability result, stated above. We present a simple computation of an ordinary two counter machine *with* zero-test instruction, a corresponding ACM computation, and a corresponding linear logic proof.

Repeating from the introduction, a key insight is that searching for a directed proof of a linear logic sequent in a theory is analogous to searching for an accepting ACM computation. A successful search is exactly an accepting computation.

Suppose the transition relation δ of a standard two counter machine with zero-test consists of the following:

$$\begin{array}{lcl} \delta_1 & ::= & Q_I \textbf{ Increment } A\ Q_2 \\ \delta_2 & ::= & Q_3 \textbf{ Decrement } A\ Q_F \\ \delta_3 & ::= & Q_2 \textbf{ Zero-Test } B\ Q_3 \end{array}$$

The machine may perform the following transitions, where an instantaneous description of a two counter machine is given by the triple consisting of Q_j, the current state, and the values of counters A and B.

$$\langle Q_I, 0, 0\rangle \xrightarrow{\delta_1} \langle Q_2, 1, 0\rangle \xrightarrow{\delta_3} \langle Q_3, 1, 0\rangle \xrightarrow{\delta_2} \langle Q_F, 0, 0\rangle$$

This computation starts in state Q_I, increments the A counter and steps to state Q_2. Then it tests the B counter for zero, and moves to Q_3, where it then decrements the A counter, moves to Q_F, and accepts.

The transition relation δ may be transformed into a transition relation δ' for an equivalent and-branching two counter machine *without* zero-test. The modified relation δ' (shown on the left below), may then be encoded as a linear logic theory (shown on the right):

		Transitions	*Theory Axioms*
δ'_1	$::=$	Q_I **Increment** A Q_2	$\vdash q_I^\perp, (q_2 \otimes a)$
δ'_2	$::=$	Q_3 **Decrement** A Q_F	$\vdash q_3^\perp, a^\perp, q_F$
δ'_3	$::=$	Q_2 **Fork** Z_B, Q_3	$\vdash q_2^\perp, (z_B \oplus q_3)$
δ'_4	$::=$	Z_B **Decrement** A Z_B	$\vdash z_B^\perp, a^\perp, z_B$
δ'_5	$::=$	Z_B **Fork** Q_F, Q_F	$\vdash z_B^\perp, (q_F \oplus q_F)$

Notice how the first two transitions (δ_1 and δ_2) of the standard two counter machine are preserved in the translation from δ to δ'. Also, the Zero-Test instruction δ_3 is encoded as three ACM transitions — δ'_3, δ'_4, and δ'_5. The transition δ'_3 is a fork to a special state Z_B, and one other state, Q_3. The two extra transitions, δ'_4 and δ'_5, force the computation branch starting in state Z_B to verify that counter B is zero. Given the above transitions, the and-branching machine without zero-test may then perform these moves:

$$\{\langle Q_I, 0, 0\rangle\} \xrightarrow{\delta'_1} \{\langle Q_2, 1, 0\rangle\} \xrightarrow{\delta'_3} \{\langle Z_B, 1, 0\rangle, \langle Q_3, 1, 0\rangle\} \xrightarrow{\delta'_4} \{\langle Z_B, 0, 0\rangle, \langle Q_3, 1, 0\rangle\} \xrightarrow{\delta'_5}$$
$$\{\langle Q_F, 0, 0\rangle, \langle Q_F, 0, 0\rangle, \langle Q_3, 1, 0\rangle\} \xrightarrow{\delta'_2} \{\langle Q_F, 0, 0\rangle, \langle Q_F, 0, 0\rangle, \langle Q_F, 0, 0\rangle\}$$

Note that an instantaneous description of this and-branching machine is a list of triples, and the machine accepts if and only if it is able to reach $\langle Q_F, 0, 0\rangle$ in *all* branches of its computation. This particular computation starts in state Q_I, increments the A counter and steps to state Q_2. Then it forks into two separate computations; one which verifies that the B counter is zero, and the other which proceeds to state Q_3. The B counter is zero, so the proof of that branch proceeds by decrementing the A counter to zero, and jumping to the final state Q_F. The other branch from state Q_3 simply decrements A and moves to Q_F. Thus all branches of the computation terminate in the final state with both counters at zero, resulting in an accepting computation.

The linear logic proof corresponding to this computation is displayed in Figures 3.1 and 3.2, and is explained in the following paragraphs. In these proofs, each application of a theory axiom corresponds to one step of ACM computation. We represent the values of the ACM counters in unary by copies of the formulas $a^\perp$ and $b^\perp$. In this example the B counter is always zero, so there are no occurrences of $b^\perp$.

The proof shown in Figure 3.1 of $\vdash z_B^\perp, a^\perp, q_F$ in the above linear logic theory corresponds to the ACM verifying that the B counter is zero. Reading the proof bottom up, it begins with a directed cut. The sequent $\vdash z_B^\perp, q_F$ is left as an intermediate step. The next step is to use another directed cut, and after application of the & rule, we have two sequents left to prove: $\vdash q_F^\perp, q_F$ and $\vdash q_F^\perp, q_F$. Both of these correspond to the ACM triple $\langle Q_F, 0, 0\rangle$ which is the accepting triple, and are provable by the identity rule. If we had attempted to prove this sequent with some occurrences of $b^\perp$, we would be unable to complete the proof.

$$\dfrac{\dfrac{}{\vdash z_B^\perp, a^\perp, z_B}\,\delta'_4 \qquad \dfrac{\dfrac{}{\vdash z_B^\perp, (q_F \oplus q_F)}\,\delta'_5 \qquad \dfrac{\dfrac{}{\vdash q_F^\perp, q_F}\,\mathbf{I} \quad \dfrac{}{\vdash q_F^\perp, q_F}\,\mathbf{I}}{\vdash (q_F^\perp \,\&\, q_F^\perp), q_F}\,\&}{\vdash z_B^\perp, q_F}\,\mathbf{Cut}}{\vdash z_B^\perp, a^\perp, q_F}\,\mathbf{Cut}$$

Figure 3.1: Zero-test proof

$$\dfrac{\dfrac{}{\vdash q_I^\perp, (q_2 \otimes a)}\,\delta'_1 \qquad \dfrac{\dfrac{\dfrac{}{\vdash q_2^\perp, (z_B \oplus q_3)}\,\delta'_3 \qquad \dfrac{\dfrac{\vdots}{\vdash z_B^\perp, a^\perp, q_F} \qquad \dfrac{\dfrac{}{\vdash q_3^\perp, a^\perp, q_F}\,\delta'_2 \quad \dfrac{}{\vdash q_F^\perp, q_F}\,\mathbf{I}}{\vdash q_3^\perp, a^\perp, q_F}\,\mathbf{Cut}}{\vdash (z_B^\perp \,\&\, q_3^\perp), a^\perp, q_F}\,\&}{\vdash q_2^\perp, a^\perp, q_F}\,\mathbf{Cut}}{\vdash (q_2^\perp \parr a^\perp), q_F}\,\parr}{\vdash q_I^\perp, q_F}\,\mathbf{Cut}$$

Figure 3.2: Proof corresponding to computation

The proof shown in Figure 3.2 of $\vdash q_I^\perp, q_F$ in the same theory demonstrates the remainder of the ACM machinery. The lowermost introduction of a theory axiom, cut, and $\parr$ together correspond to the application of the increment instruction δ'_1. That is, the $q_I^\perp$ has been "traded in" for $q_2^\perp$ along with $a^\perp$. The application of a directed cut and & correspond to the fork instruction, δ'_3 which requires that both branches of the proof be successful in the same way that and-branching machines require all branches to reach an accepting configuration. The elided proof of $\vdash z_B^\perp, a^\perp, q_F$ appears in Figure 3.1, and corresponds to the verification that the B counter is zero. The application of cut and identity correspond to the final decrement instruction of the computation, and complete the proof.

Chapter 4

Noncommutative Propositional Linear Logic

The following may be called the *unrestricted exchange* rule:

$$\mathbf{E} \quad \frac{\vdash \Sigma, \Gamma, A}{\vdash \Sigma, A, \Gamma}$$

Since sequents are treated as *multisets* of formulas in linear logic, the **E** rule is implicitly present in full linear logic. This structural rule allows sequents to be permuted arbitrarily, making linear logic a commutative logic. More specifically, $\vdash (A \otimes B) \multimap (B \otimes A)$ is derivable in linear logic using exchange, as are the analogous sequents for all the other binary connectives of linear logic ($\parr$, $\oplus$, $\&$). Since sequents are considered to be implicitly commutative, the **E** rule does not explicitly appear in proofs or lists of proof rules for linear logic. However, the absence of the **E** rule (treating sequents as *lists* of formulas) drasticly alters the set of provable sequents in linear logic. In fact, without the exchange rule, $\vdash (A \otimes B) \multimap (B \otimes A)$ is not derivable. Thus the **E** rule forces $\otimes$ (and other connectives) to be commutative.

Noncommutative propositional linear logic is linear logic where the unrestricted exchange rule is omitted, or equivalently, where sequents are treated as being lists instead of multisets. This entire family of logics is quite speculative, ad hoc, and most formulations are original to this paper. Thus one should not take too seriously any of the results of this chapter.

The family of noncommutative linear logics may be derived from linear logic by treating sequents as *lists* of formulas, instead of multisets. Thus the order of formulas in a sequent becomes important. However, the immediate resulting system is unsatisfying in that the reusable formulas (those marked by ?) are exactly the ones which can be contracted and weakened in linear logic, and thus should be permitted the freedom of exchange, even in the noncommutative versions of linear logic.

There are a whole family of logics which could result from various additions of restricted exchange to noncommutative linear logic. The main point of difference within this family is the exact formulation of the rules of inference. However, most members of this family of logics have an undecidable provability problem.

In fact, the multiplicative and reuse operators are sufficient to encode undecidable problems in most of these logics. In other words, the constants and additive connectives are not necessary in order to simulate a Turing machine in noncommutative linear logic, although they appear to be necessary in (commutative) linear logic. Below we present the detailed proof of undecidability for a particular logic we will call NCL, which is actually the multiplicative and reuse fragment of a member of the noncommutative linear logic family.

David Yetter [Yet90] has also studied a variant of noncommutative linear logic. In his work, he considered a system with two new modalities, k and K, which are related to ? and !. The k modality essentially marks those formulas which are free to be permuted, despite the noncommutativity of the logic in general. The reusable formulas (marked with ?) are allowed to permute, but are also allowed the freedom of contraction and weakening, while the k and K formulas are not. We find no compelling reason for these extra connectives, except to facilitate the encoding of (commutative) linear logic in noncommutative linear logic by prefixing every subformula with k or K. Based on our results below, other encodings of linear logic into noncommutative linear logic without k and K exist. Thus we do not include any new connectives or modalities in our logics presented below, and allow only the reusable ? formulas to permute.

We now focus on one particular member of the family of noncommutative linear logics. This logic will include only the multiplicative and reuse connectives of linear logic, excluding the additives. In order to mesh smoothly with the earlier chapters we will use the proof rules as presented in Appendix B, but add two new rules, one which allows ? formulas to permute, and one which allows an entire sequent to be "rotated". Thus one may think of a sequent as a circular list of formulas. We call this logic NCL for "circular logic".

4.1 NCL Proof Rules

The system NCL includes the **I, Cut,** $\otimes$, $\parr$, $?W, ?C, ?D$, and $!S$ rules of linear logic, with each sequent treated as a *circular* list of formulas instead of as a multiset. The following two rules of inference are also included in NCL:

$$?\mathbf{E}\quad \frac{\vdash \Gamma, \Sigma, ?A}{\vdash \Gamma, ?A, \Sigma} \qquad\qquad \frac{\vdash \Gamma, A}{\vdash A, \Gamma}\quad \mathbf{R}$$

These rules allow one to permute and exchange the reusable (?) formulas arbitrarily, and to *rotate* the entire sequent of reusable and non-reusable formulas. As our

previous discussions of linear logic accepted exchange as "part of the system" by considering sequents to be *multisets* of formulas, we will now consider the exchange rules ?**E** and **R** implicitly, by regarding sequents as circular lists of linear logic formulas.

4.2 NCL is Undecidable

We will show the word problem for semi-Thue systems has a straightforward encoding in NCL. Since we have already shown that full linear logic is undecidable, the fact that full noncommutative linear logic is undecidable is not too surprising. But since NCL is a fragment of noncommutative linear logic which does not contain the additive connectives, the earlier construction of and-branching two-counter machines in full linear logic would fail in NCL. However, the and-branching used in that construction was required in order to encode zero-test in a commutative setting. In a noncommutative setting a zero test operation may be encoded easily without any sort of and-branching. This situation is analogous to that for commutative versus noncommutative semi-Thue systems, where the noncommutative version allows the encoding of a zero test leading to undecidability, whereas the commutative version is unable to simulate zero test and has been shown to be decidable [Kos82]. In fact, since NCL closely resembles semi-Thue systems, we will demonstrate undecidability of NCL by a reduction from semi-Thue systems.

Although the reduction is simple, the proof of its correctness requires some elaborate machinery. In particular the proof of cut-elimination given in Appendix A has been given in such a way that it applies to NCL as well as linear logic.

Lemma 4.1 (Cut Elimination Revisited) *If there is a proof of sequent* $\vdash \Gamma$ *in* NCL*, then there is a cut-free proof of* $\vdash \Gamma$ *in* NCL*.*

Proof. The cut-elimination theorem for full linear logic in Appendix A is presented in a way which gives a cut elimination procedure for NCL. The only violations of the list-ordering of rules are those which correspond to the permutation of reusable formulas, such as in the case of principal cuts of $\otimes$ versus !**S**. However, these cases are legal NCL cut proof steps, since we are assuming the NCL proof rules ?**E** and **R** are *built in* to the NCL system. Thus the cut-elimination procedure for full linear logic may be employed to remove cuts from NCL proofs. ∎

Corollary 4.2 (Subformula Property Revisited) *Any formula in any cut-free* NCL *proof of* $\vdash \Gamma$ *is a subformula of* Γ*.*

Proof. Immediate. ∎

4.3 NCL Theories

We will define theories as for the commutative case (see Section 3.2), and show that cut-standardization (see Lemma 3.1) again holds in this logic.

Formally, a NCL *axiom* may be any NCL sequent of the form $\vdash C, p_{i_1}^{\perp}, p_{i_2}^{\perp}, ..., p_{i_n}^{\perp}$, where C is any NCL formula not including modal operators (? or !), and the remainder of the sequent is made up of negative literals. Any finite set of NCL axioms is a NCL *theory*. For any theory T, we say that a sequent $\vdash \Gamma$ is provable in T exactly when we are able to derive $\vdash \Gamma$ using the standard set of NCL proof rules and NCL axioms from T. Thus each axiom of T is treated as a reusable sequent which may occur as a leaf of a proof tree. As before we will write $\overline{\vdash \Gamma}^{\mathbf{T}}$ for a leaf sequent which is a member of the theory T.

We recall the definition of a *directed* cut: a *directed* cut is one where at least one premise is an axiom $\vdash C, p_{i_1}^{\perp}, p_{i_2}^{\perp}, ..., p_{i_n}^{\perp}$ in T, and C is the cut-formula in that axiom. We call any axiom premise of a directed cut where the cut-formula in that axiom is not a negative literal a *principal axiom* of that directed cut. By definition, all directed cuts have at least one principal axiom. A cut between two axioms is always directed, and if the cut-formula of such a cut is non-atomic, that cut has two principal axioms. A *directed* or *standardized* proof is a proof with all cuts directed.

Lemma 4.3 (NCL **Cut Standardization**) *If there is a proof of* $\vdash \Gamma$ *in theory* T *in* NCL, *then there is a directed proof of* $\vdash \Gamma$ *in theory* T *in* NCL.

Proof. The proof of cut-standardization in full linear logic (Lemma 3.1) applies to this case with little modification. In fact there are fewer cases here since the constants and additive connectives are not present in NCL. ∎

The earlier translation from theories into pure linear logic is also a translation of NCL theories into pure NCL. For convenience, we repeat the definition here: The translation $[T]$ of a NCL theory T with k axioms is the list of NCL formulas:

$$[\{t_1, t_2, \cdots, t_k\}] = ?[t_1], ?[t_2], \cdots, ?[t_k]$$

where $[t_i]$ is defined for each axiom t_i as follows:

$$[\vdash C, p_a^{\perp}, p_b^{\perp}, ..., p_z^{\perp}] \triangleq (C \parr p_a^{\perp} \parr p_b^{\perp} \parr \cdots \parr p_z^{\perp})^{\perp} \;=\; (p_z \otimes \cdots \otimes p_b \otimes p_a \otimes C^{\perp})$$

Note that the $p_j^{\perp}$'s are negative literals.

Lemma 4.4 (NCL **Theory** $\Rightarrow$) *For any finite set of axioms* T, $\vdash \Gamma$ *is provable in theory* T *only if* $\vdash [T], \Gamma$ *is provable.*

Proof. The proof of Lemma 3.2 carries over to NCL without modification, since the ?**E** and **R** rules are considered implicit. ∎

Lemma 4.5 (NCL **Theory** $\Leftarrow$) *For any finite set of axioms T, $\vdash \Gamma$ is provable in theory T if $\vdash [T], \Gamma$ is provable.*

Proof. Similarly, the proof of Lemma $\Leftarrow$ 3.3 given earlier applies here without modification in the system where the $?\mathbf{E}, \mathbf{R}$ rules are considered part of the NCL system. ∎

4.3.1 Semi-Thue Systems

A *semi-Thue system* $\mathcal{T}$ over alphabet Σ is a set of pairs $\langle x \to y \rangle$, where x and y are strings over Σ. Each pair in $\mathcal{T}$ is called a *production*, and we use them as rewrite rules. We call x the *left hand side* and y the *right hand side* of a production $\langle x \to y \rangle$. U *rewrites* to V in system $\mathcal{T}$ with a production $\langle g \to g' \rangle$ if U and V are words over Σ, and there exist possibly null words r and s such that $U = rgs$ and $V = rg's$. We write

$$U \Longrightarrow V$$

if U rewrites to V using some production. We use the notation

$$U \Longrightarrow^* V$$

if there exists a (possibly empty) sequence of words $U_1, \cdots, U_n$ such that

$$U \Longrightarrow U_1 \Longrightarrow U_2 \Longrightarrow \cdots \Longrightarrow U_n \Longrightarrow V$$

The *word problem* for a semi-Thue system $\mathcal{T}$ the problem of determining, for a given pair of words U and V, whether or not $U \Longrightarrow^* V$ in system $\mathcal{T}$. This problem is known to be undecidable [Pos47]. The problem remains undecidable if we add the condition that V be a singleton (word of length one) n such that n does not occur in U or in the right hand side of any production, and only appears as a singleton on the left hand side of productions. This restriction is analogous to requiring that a Turing machine have a unique final state without any outgoing transitions. The semi-Thue word problem also remains undecidable if there is a special end marker symbol m which is preserved by any rules which involve it except rules involving n. That is, the initial word U begins with a symbol m which does not occur anywhere else in U, and every rule in which m appears is of the form $\langle mw_i \to mw_j \rangle$ or is of the form $\langle mw_i \to n \rangle$ for some words w_i and w_j not containing m. This restriction is analogous to requiring that a Turing machine have only a one-way infinite tape.

To show that the above restrictions preserve undecidability it suffices to give a transformation from a general word problem to a word problem of the restricted class. Specifically, given a word problem from U to V, with set of productions $\mathcal{T}$, we may add the production $mV \to n$ where m and n are new symbols which are added to Σ. We then ask the new word problem from mU to n (with the new and

the original productions), in the alphabet $\Sigma \cup \{m, n\}$. This problem is solvable if and only if the original problem is. However this new problem is of the restricted class defined above.

4.4 From Semi-Thue Systems to Noncommutative Linear Logic

We overload the definition of translation [ ] to include the case of words — the translation of a word $[ab \cdots z]$ is the list of NCL formulas $p_a^\perp, p_b^\perp, \cdots, p_z^\perp$. We also define $[ab \cdots z]^\perp$ to be the NCL formula $p_z \otimes \cdots \otimes p_b \otimes p_a$. Finally, as a notational convenience, we let $\gamma(\Gamma)$ designate ambiguously any formula which could be derived from Γ by applications of the $\parr$ rule. In other words, $\gamma(\Gamma)$ is the result of replacing some number of commas separating formulas in Γ by $\parr$.

Given a Semi-Thue system $\mathcal{T} = \{\langle a_1 \to b_1\rangle, \langle a_2 \to b_2\rangle \cdots \langle a_k \to b_k\rangle\}$, we define the NCL theory derived from $\mathcal{T}$ as the following set of sequents:

$$\begin{array}{c} \vdash [a_1], [b_1]^\perp \\ \vdash [a_2], [b_2]^\perp \\ \vdots \\ \vdash [a_k], [b_k]^\perp \end{array}$$

For a word problem P consisting of the pair U, V we define the translation $\tau(P)$ of this problem into a NCL sequent as:

$$\vdash [U], [V]^\perp$$

Now, we show that the word problem P is solvable within system $\mathcal{T}$ if and only if the translation $\tau(P)$ is provable in the theory derived from $\mathcal{T}$. We state the two parts of the equivalence as Lemmas 4.6 and 4.7.

Lemma 4.6 *The word problem P is solvable in the semi-Thue system $\mathcal{T}$ only if $\tau(P)$ is provable in the theory derived from $\mathcal{T}$.*

Proof. We proceed by induction on the length of the derivation of $U \Longrightarrow^* V$. If the derivation is trivial, that is $U \equiv V$, then we must show that the sequent

$$\vdash [V], [V]^\perp$$

is provable. Since we assume the word problem has the restricted form with V a singleton, this sequent actually has the form $\vdash [n], [n]^\perp$, which by definition of notation is $\vdash p_n^\perp, p_n$. This is provable by identity.

Suppose the derivation of $U \Longrightarrow^* V$ is a nonempty sequence:

$$U \Longrightarrow U_1 \Longrightarrow U_2 \Longrightarrow \cdots \Longrightarrow U_n \Longrightarrow V.$$

Since $U \Longrightarrow U_1$, there is some rule $\langle g \to g' \rangle$ in $\mathcal{T}$, and possibly null words r and s such that

$$\begin{array}{rclcl} [U] & = & [rgs] & = & p_{r_1}^{\perp} \cdots p_{r_i}^{\perp} p_{g_1}^{\perp} \cdots p_{g_j}^{\perp} p_{s_1}^{\perp} \cdots p_{s_k}^{\perp} \\ [U]^{\perp} & = & [rgs]^{\perp} & = & [s]^{\perp}[g]^{\perp}[r]^{\perp} \\ [U_1] & = & [rg's] & = & p_{r_1}^{\perp} \cdots p_{r_i}^{\perp} p_{g'_1}^{\perp} \cdots p_{g'_h}^{\perp} p_{s_1}^{\perp} \cdots p_{s_k}^{\perp} \\ [U_1]^{\perp} & = & [rg's]^{\perp} & = & [s]^{\perp}[g']^{\perp}[r]^{\perp} \end{array}$$

By induction we assume that we have a proof of $\vdash [U_1], [V]^{\perp}$, and construct from this a proof of the following form:

$$\dfrac{\dfrac{}{\vdash [g], [g']^{\perp}}\mathbf{T} \quad \dfrac{\dfrac{\dfrac{\vdots}{\vdash [r], [g'], [s], [V]^{\perp}}}{\vdash [r], \gamma([g']), [s], [V]^{\perp}}\parr \quad \vdots}{\vdots}\parr}{\vdash [r], [g], [s], [V]^{\perp}}\mathbf{Cut}$$

In this partial deduction there are as many applications of the $\parr$ rule as there are separate formulas in $[g']$. This is because each application of $\parr$ replaces a comma by a $\parr$ in the $\gamma([g'])$ formula.

The following concrete example illustrates the intended simulation. Assume that the first rule applied in the sequence of reductions is $\langle cd \to xy \rangle$. Then g in the above schema is cd, and g' is xy. Also, $[g]$ is $p_c^{\perp}, p_d^{\perp}$, $[g']$ is $p_x^{\perp}, p_y^{\perp}$, and $\gamma([g'])$ is $(p_x^{\perp} \parr p_y^{\perp})$. Also assume that r is mab, s is ef, and $[V]^{\perp}$ is p_n.

$$\dfrac{\dfrac{}{\vdash p_c^{\perp}, p_d^{\perp}, (p_y \otimes p_x)}\mathbf{T} \quad \dfrac{\dfrac{\vdots}{\vdash p_m^{\perp}, p_a^{\perp}, p_b^{\perp}, p_x^{\perp}, p_y^{\perp}, p_e^{\perp}, p_f^{\perp}, p_n}}{\vdash p_m^{\perp}, p_a^{\perp}, p_b^{\perp}, (p_x^{\perp} \parr p_y^{\perp}), p_e^{\perp}, p_f^{\perp}, p_n}\parr}{\vdash p_m^{\perp}, p_a^{\perp}, p_b^{\perp}, p_c^{\perp}, p_d^{\perp}, p_e^{\perp}, p_f^{\perp}, p_n}\mathbf{Cut}$$

Thus, by induction, given a sequence of reductions which solves a word problem, we may simulate the solution in NCL. ∎

Lemma 4.7 *The word problem P is solvable with productions $\mathcal{T}$ if $\tau(P)$ is provable in the theory derived from $\mathcal{T}$.*

Proof. We remember that the target word V of the word problem is a singleton and that there is a special marker symbol m at the beginning of U which is preserved by all the productions in $\mathcal{T}$.

The construction of a rewrite sequence begins with any proof of $\tau(P)$, and then produces a directed proof by cut-elimination. We may read any directed proof of $\tau(P)$ as a solution to the word problem P, since each directed cut corresponds directly to the application of one reduction in the semi-Thue system.

In more detail, given a proof which ends in a sequent $\vdash \Gamma$ where $\vdash \Gamma$ is equal to $\tau(P)$, possibly with some of the commas in $\tau(P)$ replaced with $\parr$, we apply the NCL cut-standardization Lemma 4.3, and obtain a directed proof of $\vdash \Gamma$. We now prove by induction on the length of the directed proof that this proof may be mimicked by the corresponding semi-Thue system.

In the case that $\vdash \Gamma$ is equal to $\vdash [V], [V]^{\perp}$, (for example, this proof may be a single application of the identity rule) then the solution to the word problem is trivial, i.e., no productions are required, since $U \equiv V$.

In the induction step, $\vdash \Gamma$ cannot be provable with the identity rule. Inspecting the other rules of NCL, we see that $\otimes$ and the ? and ! rules are inapplicable, since the conclusion sequent does not contain any occurrences of $\otimes$, ?, or !. If $\parr$ is the last rule applied in the proof, then we appeal to the induction hypothesis, since then we simply have replaced fewer commas with $\parr$. The only remaining case is **Cut**, and by cut-standardization, we know one hypothesis is an axiom, and the cut-formula in that axiom is not a negative literal. Inspecting the axioms in the theory corresponding to the productions of a semi-Thue system, we see that the cut-formula must always be a formula $[g']^{\perp}$. This formula is built up from positive literals connected by $\otimes$. The cut-formula in the other hypothesis then must be built from negative literals connected by $\parr$.

Since the start marker m on the initial word U is preserved by each production in $\mathcal{T}$, we know that the NCL theory derived from $\mathcal{T}$ has a special property: the translation p_m of m may not be a subformula of the cut-formula of a directed cut unless it also appears in the first non-principal position following the cut-formula in the axiom. Thus the start marker participates in a directed cut if and only if it is the first symbol in the sequence of symbols replaced by the cut. It follows that the translation p_m of the start marker is preserved in any directed proof of any well formed word problem. This allows us to conclude that no rule is applied "around the end" of a word through some convoluted use of the rotation rule, and thus that the semi-Thue system may mimic the NCL proof by applying the corresponding production to the string. By induction, we have our result. ∎

Theorem 4.8 *The provability problem for* NCL *is recursively unsolvable.*

Corollary 4.9 *The provability problem for* NCL *augmented with the additive (& and $\oplus$) and constant ($\perp$, 1, and $\top$) rules is recursively unsolvable.*

This corollary follows from the theorem by a conservativity result which is easily derived from the cut-elimination and subformula properties of NCL.

4.5 Other Noncommutative Logics

As mentioned previously, there is a family of logics which share a strong resemblance to NCL. All of the ones we can sensibly imagine have undecidable decision problems.

In all formulations of noncommutative linear logic the key rule is $\otimes$. In a sense which we make more precise later, the constants and additive connectives of linear logic are inherently commutative. Also, the $\parr$ rule follows the $\otimes$ rule in its commutativity. Thus noncommutative linear logic is quite sensitive to the exact formulation of $\otimes$. However, there are some minor variations on the syntactic presentation of the other proof rules which first bear some notice.

4.5.1 Rotate Rule versus Embedding

One motivation for the particular formulation of NCL studied in the previous section (in particular, the introduction of the **R** rule) is so that we may make use of the same formulation of linear logic rules given in Appendix B, and refer to the previously demonstrated lemmas about linear logic with as little modification as possible. Without the **R** rule we would have to modify the formulation of other rules, such as the $\parr$ rule, to allow its application within a sequent, instead of requiring its application at one end of a sequent. To see this, compare the original version of $\parr$ on the left with the modified version on the right below:

$$\parr \quad \frac{\vdash \Sigma, A, B}{\vdash \Sigma, (A \parr B)} \qquad \frac{\vdash \Sigma, A, B, \Gamma}{\vdash \Sigma, (A \parr B), \Gamma} \quad \parr\mathbf{2}$$

We will call the $\parr\mathbf{2}$ rule the embedded equivalent of the $\parr$ rule. The use of $\parr\mathbf{2}$ in noncommutative linear logic without the **R** rule directly corresponds to the use of $\parr$ in NCL with **R** rule considered part of the system. We will use ENCL to stand for the system derived from NCL by removing the **R** rule, and replacing all other rules by their embedded equivalents, and adding a symmetric identity rule.

Lemma 4.10 *A sequent* $\vdash \Gamma$ *is provable in* NCL *if and only if it is provable in* ENCL.

This lemma follows by induction on the length of proofs, and from this lemma we obtain undecidability for this system.

Corollary 4.11 *The provability problem for* NCL *without the* **R** *rule, and with every other rule replaced by its embedded equivalent is recursively unsolvable.*

4.5.2 NCL without ?E

The earlier proof of undecidability fails in NCL without the ?**E** rule, since some of the requisite lemmas about theories fail. However, we may omit this rule, and replace **?C** with the following **?C2** rule to restore our results, and many other properties of noncommutative linear logic.

$$\textbf{?C}\quad \frac{\vdash \Sigma, ?A, ?A}{\vdash \Sigma, ?A} \qquad\qquad \frac{\vdash \Sigma, ?A, \Gamma, ?A}{\vdash \Sigma, \Gamma, ?A}\quad \textbf{?C2}$$

This contraction rule essentially states that what may be proven from two *not necessarily contiguous* assumptions of a reusable formula, may be proven from one assumption of that reusable formula. It is the case that the **?C2** rule is derivable from the **?E** and **?C** rules in NCL.

Lemma 4.12 *A sequent* $\vdash \Gamma$ *is provable in* NCL *if and only if it is provable in* NCL *without the* **?E** *rule, and with the* **?C** *rule replaced by* **?C2**.

This lemma may be proven by induction on the length of proofs. Essentially, in NCL one may contract and then exchange the reusable formula to any desired position, while in the other system one may contract the formula directly into position. On the other hand, to permute a reusable formula in NCL, one simply applies exchange, while in the other system one must contract the formula into position, and then weaken away the formula in its previous position. Using this lemma, we may obtain the following undecidability result.

Corollary 4.13 *The provability problem for* NCL *without the* **?E** *rule, and with the* **?C** *rule replaced by* **?C2** *rule is recursively unsolvable.*

4.5.3 Alternate $\otimes$

There are two quite reasonable versions of the $\otimes$ rule in noncommutative linear logic, one as used above defined in Appendix B, and the other using a different sequent order in the conclusion:

$$\otimes\quad \frac{\vdash \Sigma, A \qquad \vdash B, \Gamma}{\vdash \Sigma, (A \otimes B), \Gamma} \qquad\qquad \frac{\vdash \Sigma, A \qquad \vdash \Gamma, B}{\vdash \Sigma, \Gamma, (A \otimes B)}\quad \otimes\mathbf{2}$$

The two formulations are equivalent in the presence of unrestricted exchange (commutative linear logic), but are subtlely different in the context of noncommutative linear logic. In a noncommutative linear logic with $\otimes$ replaced by $\otimes\mathbf{2}$, the definition of negation must change. In particular, the negation of the multiplicative connectives would be defined as follows:

$$(A \otimes B)^{\perp} \triangleq A^{\perp} \parr B^{\perp} \qquad\qquad (A \parr B)^{\perp} \triangleq A^{\perp} \otimes B^{\perp}$$

We define the translation $\sigma(\Gamma)$ of a sequent Γ to be the sequent Γ with all occurrences of formulas of the form $A \otimes B$ replaced by $B \otimes A$.

Lemma 4.14 *A sequent $\vdash \Gamma$ is provable in* NCL *if and only if $\sigma(\Gamma)$ is provable in* NCL *with the $\otimes$ rule replaced by the $\otimes\mathbf{2}$ rule*

Lemma 4.15 *A sequent $\vdash \sigma(\Gamma)$ is provable in* NCL *if and only if Γ is provable in* NCL *with the $\otimes$ rule replaced by the $\otimes\mathbf{2}$ rule*

This lemma follows by induction on the height of proofs.

Lemma 4.16 *The provability problem for* NCL *with the $\otimes$ rule replaced by the $\otimes\mathbf{2}$ rule and alternate definition of negation is recursively unsolvable.*

This lemma follows from the above two, which simply state that by reversing the order of all tensor ($\otimes$) formulas, we pass from NCL to this new logic, and back again. Thus a decision procedure for one implies a decision procedure for the other, and by Lemma 4.8, we know there is no decision procedure for NCL.

4.5.4 Mix and Match

The above modifications of NCL do not interfere with cut-elimination, nor with the basic undecidability result for NCL. It is also the case that even in combination the above three modifications, ($\mathbf{R}$ versus embedding, $?\mathbf{E}$ versus $?\mathbf{C2}$, and $\otimes$ versus $\otimes\mathbf{2}$) do not interact. That is, any combination of these modications retains the character of NCL, including the properties of being undecidable, and having a cut-elimination theorem.

4.6 Degenerate Noncommutative Linear Logics

Some variations on the NCL system are not as benign as the above. In fact, it is much easier to create nonsense than a coherent logic by altering proof rules haphazardly.

The main focus of this section is to consider plausible but degenerate variants of the rules based on interleaving the circular orders of hypotheses.

4.6.1 Intermingling $\otimes$

At first glance, it might seem interesting to study the systems obtained when binary rules in NCL are replaced with rules which allow intermingling of the hypotheses in the conclusion. For example,

$$\otimes\mathbf{3} \quad \frac{\vdash \Sigma, A \qquad \vdash B, \Gamma}{\vdash \Delta, (A \otimes B)} \quad \text{where } \Delta \text{ is some interleaving of } \Sigma \text{ and } \Gamma$$

Somewhat surprisingly, the system obtained by replacing $\otimes$ with $\otimes\mathbf{3}$ in NCL is equivalent to a commutative version of NCL.

Lemma 4.17 *A sequent $\vdash \Gamma$ is provable in the system obtained by replacing $\otimes$ by $\otimes\mathbf{3}$ in* NCL *if and only if that sequent is provable in the system obtained by adding the unrestricted exchange rule to* NCL.

This lemma follows by induction on the length of cut-free proofs. Formally, we need a cut-elimination procedure for both logics. The cut-elimination procedure for full linear logic suffices to eliminate cuts from NCL with unrestricted exchange. Cut-elimination for NCL with $\otimes$ replaced by $\otimes\mathbf{3}$ is possible to prove directly, although the principal $\otimes\mathbf{3}$ versus $\parr$ case is quite difficult. Cut-elimination in this case may be accomplished with the addition of an "intermingling cut" rule which along with the nonintermingling cut rule may be eliminated from any proof. The key reason this lemma holds is that $\otimes$ and $\parr$ are the only binary connectives of NCL and allowing $(A \otimes B)$ to be equivalent to $(B \otimes A)$ in this context causes $(A \parr B)$ to be equivalent to $(B \parr A)$.

Corollary 4.18 *A sequent $\vdash \Gamma$ is provable in the system obtained by replacing $\otimes$ by $\otimes\mathbf{3}$ in* NCL *augmented with additives and constants if and only if that sequent is provable in the system obtained by adding the unrestricted exchange rule to* NCL *augmented with additives and constants.*

This corollary follows from the fact that the constants and additive connectives are inherently commutative, and may be proven by induction on the length of proofs.

4.6.2 Intermingling Cut

A problem similar to that which occurs with $\otimes\mathbf{3}$ arises if we allow the **Cut** rule to interleave its conclusion. Define **Cut2**:

$$\textbf{Cut2} \quad \frac{\vdash \Sigma, A \qquad \vdash \Gamma, A^{\perp}}{\vdash \Delta} \quad \text{where } \Delta \text{ is some interleaving of } \Sigma \text{ and } \Gamma$$

As for the previous alteration, the system NCL with **Cut** replaced by **Cut2** would be commutative. We can achieve the effect of the unrestricted exchange rule using the **Cut2** rule:

$$\frac{\begin{array}{c}\vdots\\ \vdash \Sigma, \Gamma, A\end{array} \qquad \begin{array}{c}\vdots\\ \vdash A, A^{\perp}\end{array}}{\vdash \Sigma, A, \Gamma}\textbf{Cut2}$$

Note that for any formula A, there is always a proof of $\vdash A, A^{\perp}$ in noncommutative (as well as commutative) linear logic. The above partial deduction shows that

unrestricted exchange may be simulated in noncommutative linear logic with **Cut** replaced by **Cut2**. Somewhat more concretely, the following shows a deduction of a sequent which is not derivable in NCL:

$$\dfrac{\dfrac{\dfrac{}{\vdash p_1, p_1^\perp}\mathbf{I} \quad \dfrac{}{\vdash p_2, p_2^\perp}\mathbf{I}}{\vdash p_1, (p_1^\perp \otimes p_2^\perp), p_2}\otimes \qquad \dfrac{}{\vdash p_1, p_1^\perp}\mathbf{I}}{\vdash (p_1^\perp \otimes p_2^\perp), p_1, p_2}\mathbf{Cut2}$$

Notice in the final conclusion that p_1 and p_2 have changed places, in a way impossible without the use of the **Cut2** rule in NCL. Thus using **Cut2** we could prove any sequent which is provable in the commutative fragment of linear logic corresponding to NCL. However, it would be impossible to prove some such sequents in NCL without **Cut2**, and thus cut-elimination fails in this logic.

However, since there is a proof of a sequent in this logic if and only if there is a proof of that sequent in (commutative) linear logic, we may as well use linear logic, which does have a cut-elimination theorem.

Chapter 5

The Multiplicative Fragment

We now consider the fragment of linear logic which includes only the multiplicative connectives. In Section 5.1, we show that the decision problem for this fragment is in NP, and in Section 5.2 we show that the multiplicative fragment with a rule of unrestricted weakening is NP-complete.

5.1 Multiplicatives

We have two results which characterize the complexity of this fragment incompletely. The exact complexity of this fragment is one of the significant open problems. In short, we find that this fragment is in NP, and if we introduce the structural rule of unrestricted weakening into this fragment, it becomes NP-complete.

The pure multiplicative fragment (without additive connectives or storage operators) is the simplest fragment of linear logic that we have investigated.

Theorem 5.1 *Multiplicative linear logic is in* NP.

Proof. The proof is straightforward: Each connective in the conclusion sequent is the principal connective in exactly one proof step in any cut-free proof, thus giving a polynomial bound on the size of cut-free proofs. Thus the entire proof may be guessed in polynomial time. ∎

5.2 Direct Logic

We have been unable to prove the multiplicative fragment of linear logic NP-complete. We now believe that this may be difficult, due to the lack of redundancy in this problem statement [GJ79]. As part of our investigation of the need to discard arbitrary resources to achieve NP-completeness, we studied propositional multiplicative linear logic with unrestricted weakening, but without contraction. We will call this *direct*

logic or DL, as it is similar to the direct logic of [KW84]. DL is also considered in considerable detail in Bellin [Bel90]. The rules for this system are the identity and cut rules, the rules for the multiplicatives, constants, and the structural rule **W** of unrestricted weakening.

$$\mathbf{W}\quad \frac{\vdash \Sigma}{\vdash A, \Sigma}$$

We first demonstrate cut-elimination for DL, yielding consistency, which will facilitate our later proof of NP-completeness.

Lemma 5.2 *A sequent is provable in* DL *if and only if it is provable in* DL *without using the cut rule.*

Proof. This lemma, as the cut-elimination theorem for full linear logic, is proven by giving a cut-elimination procedure. This procedure takes any proof as input, and produces a cut-free proof of the same sequent.

We modify the procedure given in Appendix A. More specifically, we modify the procedure given in Lemma A.1 to handle the extra cases brought about by the presence of weakening, and we define the formula A in the above presentation of **W** to be the principal formula of **W**.

Thus we need to present two reductions, one in the case that the cut formula is principal, and in case it is not principal.

We will follow the notation used in Appendix A, where **Cut**$*$ is used to ambiguously refer to the **Cut** rule or the extra rule of inference introduced in the appendix called **Cut**!.

First, we consider the non-principal **W** case:

$$\dfrac{\dfrac{\begin{array}{c}\vdots\\ \vdash \Sigma, A\end{array}}{\vdash \Sigma, A, B}\mathbf{W} \qquad \begin{array}{c}\vdots\\ \vdash \Gamma, A^{\perp}\end{array}}{\vdash \Sigma, \Gamma, B}\mathbf{Cut}* \quad \Longrightarrow \quad \dfrac{\dfrac{\begin{array}{c}\vdots\\ \vdash \Sigma, A\end{array} \qquad \begin{array}{c}\vdots\\ \vdash \Gamma, A^{\perp}\end{array}}{\vdash \Sigma, \Gamma}\mathbf{Cut}*}{\vdash \Sigma, \Gamma, B}\mathbf{W}$$

The above reduction is very similar to the case of non-principal **?W**.

In the case of principal **W**, we have the following reduction:

$$\dfrac{\dfrac{\begin{array}{c}\vdots\\ \vdash \Sigma\end{array}}{\vdash \Sigma, A}\mathbf{W} \qquad \begin{array}{c}\vdots\\ \vdash \Gamma, A^{\perp}\end{array}}{\vdash \Sigma, \Gamma}\mathbf{Cut}* \quad \Longrightarrow \quad \dfrac{\begin{array}{c}\vdots\\ \underline{\vdash \Sigma}\mathbf{W}\\ \vdots\end{array}}{\vdash \Sigma, \Gamma}\mathbf{W}$$

By combining these two reductions with the procedure given in Lemma A.1, we have a cut-reduction lemma for DL.

Fortunately, Lemma A.2 and Theorem A.3 then follow without modification. Formally, we must show that although the lemmas in Appendix A apply to full linear logic, they would not take a proof in DL into a proof outside DL. By inspection of the reductions used for the subset of the connectives of DL, we see that this holds.

Therefore, given any DL proof of sequent $\vdash \Gamma$ in theory T, we can construct a cut-free proof of $\vdash \Gamma$ in theory T. ∎

Lemma 5.3 *The provability problem for* DL *is in* NP.

Proof. The membership in NP of the provability problem for DL follows from a polynomial bound on the size of cut-free DL proofs. Each subformula occurrence in the conclusion is analyzed in at most one rule application in any cut-free proof of that conclusion. Thus, given a provable sequent, it is possible to nondeterministically generate and check a cut-free proof in polynomial time. ∎

The proof of the NP-hardness of provability for DL is obtained by a transformation from the Vertex Cover problem which can be stated as: Given a graph $G = (V, E)$ and a bound k, find a subset U of k or fewer vertices from V such that every edge in E is incident on some vertex in U. Given an instance of the Vertex Cover problem, we construct $\gamma(V, E, k)$, a DL sequent which is provable exactly when (V, E) has a vertex cover of fewer than k vertices. Let $deg(v, E)$ denote the degree of vertex v, i.e., the number of edges in E incident on v. The definition of γ is given by

$$\begin{aligned}
\gamma(V,E,k) &= \vdash m^k, \phi(V,E), \epsilon(E) \\
\phi(V,E) &= \parr_{v \in V} (m^\perp \otimes (\underbrace{x_v^\perp \parr \ldots \parr x_v^\perp}_{deg(v,E)})) \\
\epsilon(E) &= \otimes_{(u,v) \in E} (x_u \parr x_v)
\end{aligned}$$

For example, given $V = \{a, b, c, d\}$ and $E = \{(a, b), (c, d), (b, c)\}$, and $k = 2$, the sequent $\gamma(V, E, k)$ is

$$\begin{aligned}
\vdash \quad & m, m, \\
& m^\perp \otimes x_a^\perp,\ m^\perp \otimes (x_b^\perp \parr x_b^\perp),\ m^\perp \otimes (x_c^\perp \parr x_c^\perp),\ m^\perp \otimes x_d^\perp, \\
& (x_a \parr x_b) \otimes (x_c \parr x_d) \otimes (x_b \parr x_c)
\end{aligned}$$

A vertex cover in this example is obviously $\{b, c\}$. The corresponding deduction in DL of $\gamma(V, E, k)$ can be constructed in stages. In the first stage, the formula encoding vertices a and d are weakened and the formulas encoding vertices b and c

are reduced as shown below.

$$
\dfrac{\dfrac{\dfrac{}{\vdash m, m^{\perp}}\mathbf{I} \qquad \dfrac{\dfrac{}{\vdash m, m^{\perp}}\mathbf{I} \qquad \dfrac{\dfrac{\dfrac{\vdots}{\vdash x_b^{\perp}, x_b^{\perp}, x_c^{\perp}, x_c^{\perp}, \epsilon(E)}}{\vdash x_b^{\perp}, x_b^{\perp}, (x_c^{\perp} \parr x_c^{\perp}), \epsilon(E)}\parr}{\vdash (x_b^{\perp} \parr x_b^{\perp}), (x_c^{\perp} \parr x_c^{\perp}), \epsilon(E)}\parr}{\vdash m, (x_b^{\perp} \parr x_b^{\perp}), m^{\perp} \otimes (x_c \parr x_c), \epsilon(E)}\otimes}{\vdash m, m, m^{\perp} \otimes (x_b^{\perp} \parr x_b^{\perp}), m^{\perp} \otimes (x_c^{\perp} \parr x_c^{\perp}), \epsilon(E)}\otimes}{\vdash m, m, m^{\perp} \otimes x_a^{\perp}, m^{\perp} \otimes (x_b^{\perp} \parr x_b^{\perp}), m^{\perp} \otimes (x_c^{\perp} \parr x_c^{\perp}), m^{\perp} \otimes x_d^{\perp}, \epsilon(E)}\mathbf{W}
$$

The remaining subgoal in the above deduction can be proved in the next stage. The vertex literals corresponding to the vertex cover can be paired off with literals in the edge encodings to demonstrate that there is at least one vertex literal for each edge.

$$
\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash x_b^{\perp}, x_b}\mathbf{I}}{\vdash x_b^{\perp}, x_a, x_b}\mathbf{W}}{\vdash x_b^{\perp}, (x_a \parr x_b)}\parr \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{\vdash x_c^{\perp}, x_c}\mathbf{I}}{\vdash x_c^{\perp}, x_c, x_d}\mathbf{W}}{\vdash x_c^{\perp}(x_c \parr x_d)}\parr \qquad \dfrac{\dfrac{\dfrac{}{\vdash x_b^{\perp}, x_b}\mathbf{I}}{\vdash x_b^{\perp}, x_c^{\perp}, x_b, x_c}\mathbf{W}}{\vdash x_b^{\perp}, x_c^{\perp}, (x_b \parr x_c)}\parr}{\vdash x_b^{\perp}, x_c^{\perp}, x_c^{\perp}, (x_c \parr x_d) \otimes (x_b \parr x_c)}\otimes}{\vdash x_b^{\perp}, x_b^{\perp}, x_c^{\perp}, x_c^{\perp}, (x_a \parr x_b) \otimes (x_c \parr x_d) \otimes (x_b \parr x_c)}\otimes
$$

The next three lemmas are stated without proof. They are used in the proof of Lemma 5.7 to establish that when a vertex cover for V,E, and k exists, then $\gamma(V, E, k)$ is provable.

Lemma 5.4 *If $l \leq k$, then there exists a deduction of $\vdash m^k, \Gamma$ from $\vdash m^l, \Gamma$.*

Lemma 5.5 *Given V, E, and $U \subseteq V$, let Γ be the multiset containing $deg(u, E)$ occurrences of $x_u^{\perp}$ for each u in U, and let l be $|U|$. There is a deduction of $\vdash m^l, \phi(V, E), \Delta$ from $\vdash \Gamma, \Delta$.*

Lemma 5.6 *If Γ is a multiset of literals such that for each (u, v) in E, there is a distinct occurrence of either $x_u^{\perp}$ or $x_v^{\perp}$ in Γ, then $\vdash \Gamma, \epsilon(E)$ is provable.*

Lemma 5.7 *If $G = (V, E)$ has a vertex cover U of k or fewer vertices, then $\gamma(V, E, k)$ has a* DL *proof.*

Proof. Let $l = |U|$ be the cardinality of U. By Lemma 5.4 and the definition of γ, the required conclusion, $\vdash m^k, \phi(V, E), \epsilon(E)$, can be deduced from $\vdash m^l, \phi(V, E), \epsilon(E)$. Let Γ be the multiset of literals containing $deg(u, E)$ occurrences

of $x_u^\perp$ for each u in U. By Lemma 5.5, there is a deduction of $\vdash m^l, \phi(V,E), \epsilon(E)$ from $\vdash \Gamma, \epsilon(E)$. Since $deg(u,E)$ is the number of occurrences of x_u in $\epsilon(E)$, the sequent $\vdash \Gamma, \epsilon(E)$ is provable by Lemma 5.6. ∎

Lemma 5.8 states some straightforward properties about weakening, and is given below without proof (see [Bel90], page 138). A formula occurrence is said to be *weakened* in a proof if it is the principal formula of an application of the weakening rule. The lemma essentially captures the idea that if all subformulas of a formula occurrence are weakened, then the formula itself can be weakened instead; and if even a single conjunct in a conjunction is weakened, then the entire conjunction can be weakened instead. Lemma 5.8 is used repeatedly in the proof of Theorem 5.9 to maximize the size of any formulas that are weakened in a proof.

Lemma 5.8 *For any proof π in* DL *of a sequent $\vdash \Gamma$, one can obtain a proof θ of $\vdash \Gamma$ such that*

1. *for any formula occurrence $(A \otimes B)$, neither A nor B is weakened in θ*
2. *for any formula occurrence $(A \parr B)$, A and B are not both weakened in θ, and*
3. *any weakening of formula occurrences in Γ occurs below any application of non-weakening rules in θ.*

Lemma 5.9 *Given a graph $G = (V,E)$ and a bound k, if $\gamma(V,E,k)$ is provable in* DL *then G has a vertex cover of size less than k.*

Proof. Given a proof of $\gamma(V,E,k)$, first take the set U of vertices u such that $\vdash x_u, x_u^\perp$ is an axiom in the proof of $\gamma(V,E,k)$. There are two possible ways in which U might not be a vertex cover. One way is if for some edge encoded by $(x_v \parr x_w)$, neither x_v nor x_w appears in an axiom. Then the literals x_u and x_v must have been subformulas of some weakened formulas. By Lemma 5.8 the given proof can be transformed to one in which both x_u and x_v are not weakened, and neither is $(x_u \parr x_v)$ since it is a conjunct. Therefore the entire edge encoding $\epsilon(E)$ would have to be weakened below any non-weakening rules, and as a result $\vdash m^k, \phi(V,E)$ would have to be provable. Since $\vdash m^k, \phi(V,E)$ contains no positive occurrences of literals x_v for v in V, a proof of $\vdash m^k, \phi(V,E)$ cannot contain axioms of the form $\vdash x_v, x_v^\perp$. Again, by Lemma 5.8, each formula in $\phi(V,E)$ must be weakened below any application of logical rules. Such a proof would contain a deduction of $\vdash m^k$. However, $\vdash m^k$ is unprovable for any k, contradicting the assumption that $\vdash \gamma(V,E,k)$ is provable.

The only remaining way in which the set U with $l = |U|$ might fail to be a vertex cover is if $l > k$. The negative literals $x_v^\perp$ in axioms $\vdash x_v, x_v^\perp$ only occur in the formulas in $\phi(V,E)$. By Lemma 5.8, the given proof can be transformed to a proof θ in which l of the formulas in $\phi(V,E)$ are not weakened, because each formula in

$\phi(V, E)$ contributes at most one vertex to the set U. Since each unweakened formula in $\phi(V, E)$ is of the form $m^{\perp} \otimes A$ for some A, Lemma 5.8 implies that θ contains at least l axioms of the form $\vdash m, m^{\perp}$. However, there are only k positive occurrences of the literal m in the conclusion sequent $\vdash \gamma(V, E, k)$, and each occurrence can appear in at most one axiom of the form $\vdash m, m^{\perp}$, thus contradicting the claim that $l > k$.

Therefore, the construction of U from a DL proof of $\vdash \gamma(V, E, k)$ does yield a vertex cover for $G = (V, E)$ of size bounded by k. ∎

The encoding γ transforming an instance of the Vertex Cover problem to the provability of a DL sequent is clearly of polynomial complexity. Together, Lemmas 5.7 and 5.9 yield the following result.

Theorem 5.10 *Multiplicative linear logic with unrestricted weakening is* NP*-complete.*

In this reduction, weakening appears essential since an edge may be covered by selecting one endpoint or both, and weakening allows both cases to succeed. Only with additives would it be possible to encode such behavior in linear logic, and including the additives would take DL out of NP. In fact, DL with additive connectives becomes PSPACE-complete.

Chapter 6

Conclusion

We have investigated the complexity of the provability problem for several fragments of propositional linear logic. Our most significant results are that provability for full propositional linear logic is undecidable, but that provability becomes PSPACE-complete when the modal storage operator is removed. One may view these results in terms of the non-modal multiplicative-additive linear logic as the facts that provability in this logic without non-logical axioms is PSPACE-complete, and with non-logical axioms provability becomes undecidable. In fact, even if the non-logical axioms are of an extremely restricted class, the provability problem remains undecidable.

These results point out the greater complexity inherent in linear logic, when compared with classical or intuitionistic logic. This extra complexity is the price one should expect to pay in a logic as detailed, or as specific, as linear logic. In fact, we show that linear logic is a computational logic. That is, linear logic can exactly represent computations, to the point where not only is there a correspondence between derivable conclusions and machine configurations that eventually reach an accepting state, but there is an exact correspondence between (standardized) proofs and accepting computations.

We have also shown that provability for the non-commutative fragment of linear logic (even without additive connectives) is also undecidable. Finally, we show that the decision problem for the multiplicative fragment is in NP, and becomes NP-complete in the presence of unrestricted weakening.

Although we have gained some insight into the expressive power and combinatorial properties of propositional linear logic, some open problems remain. We have been unable to establish tight bounds for the multiplicative fragment or settle the decidability of the multiplicatives with the ! operator. This seems particularly difficult, since a positive solution would involve an extension of the reachability algorithm for Petri nets. The other open problem of interest to us is the decidability of various fragments of linear logic without the modal operators ? and !, and without

non-logical axioms, extended with propositional quantifiers.

Acknowledgements

We would like to thank the Stanford Center for Study of Language and Information, and the SRI Computer Science Laboratory for supportive working environments. Thanks also to Jean-Yves Girard, Alasdair Urquhart, Gian Luigi Bellin, and Grigori Mints for stimulating discussion and correspondence. Tim Winkler of the SRI Computer Science Laboratory helped with the typesetting of the $\mathcal{Y}$ connective. David Israel, Peter Neumann, and Narciso Martí-Oliet carefully read drafts of the report and provided several constructive comments.

Patrick Lincoln is supported by an AT&T Bell Laboratories graduate fellowship and by the SRI Computer Science Laboratory (summers of 1989 and 1990). John Mitchell is supported by an NSF PYI Award, matching funds from Digital Equipment Corporation, Powell Foundation, Xerox Corporation; NSF grant CCR-8814921, and Wallace F. and Lucille M. Davis Faculty Scholarship. Andre Scedrov is partially supported by NSF Grant CCR-87-05596, by ONR Grant NOOO14-88-K-0635 and by the 1987 Young Scientist Award from the Natural Sciences Association of the University of Pennsylvania. During his 1989-90 sabbatical at Stanford, Scedrov was also partially supported by Mitchell's PYI-related funds. Natarajan Shankar is supported by the SRI Computer Science Laboratory.

Bibliography

[AFG90] A. Asperti, G.-L. Ferrari, and R. Gorrieri. Implicative formulae in the 'proofs as computations' analogy. In *Proc. 17-th ACM Symp. on Principles of Programming Languages, San Francisco*, pages 59–71, January 1990.

[AP90] J.-M. Andreoli and R. Pareschi. Linear objects: Logical processes with built-in inheritance. In *Proc. 7-th International Conference on Logic Programming, Jerusalem*, May 1990.

[Asp87] A. Asperti. A logic for concurrency. Technical report, Dipartimento di Informatica, Universitá di Pisa, 1987.

[AV90] S. Abramsky and S. Vickers. Quantales, observational logic, and process semantics. Preprint, January 1990.

[Avr88] A. Avron. The semantics and proof theory of linear logic. *Theoretical Computer Science*, 57:161–184, 1988.

[Baw86] A. Bawden. Connection graphs. In *Proc. ACM Symp. on Lisp and Functional Programming*, pages 258–265, 1986.

[Bel90] G. Bellin. *Mechanizing Proof Theory: Resource-Aware Logics and Proof-Transformations to Extract Implicit Information.* PhD thesis, Stanford University, 1990.

[Cer90] S. Cerrito. A linear semantics for allowed logic programs. In *Proc. 5th IEEE Symp. on Logic in Computer Science, Philadelphia*, June 1990.

[CKS81] A.K. Chandra, D. C. Kozen, and L. J. Stockmeyer. Alternation. *Journal of the ACM*, 28(1):114–133, 1981.

[Clo86] P. Clote. On the finite containment problem for Petri nets. *Theoretical Computer Science*, 43:99–105, 1986.

[DR89] V. Danos and L. Regnier. The structure of multiplicatives. *Archive for Mathematical Logic*, 28:181–203, 1989.

[Gen69] G. Gentzen. *Collected Works. Edited by M.E. Szabo.* North-Holland, Amsterdam, 1969.

[GG89] C.A. Gunter and V. Gehlot. Nets as tensor theories. In G. De Michelis, editor, *Proc. 10-th International Conference on Application and Theory of Petri Nets, Bonn*, pages 174–191, 1989.

[GG90] V. Gehlot and C.A. Gunter. Normal process representatives. In *Proc. 5-th IEEE Symp. on Logic in Computer Science, Philadelphia*, June 1990.

[GH90] J.C. Guzman and P. Hudak. Single-threaded polymorphic lambda calculus. In *Proc. 5-th IEEE Symp. on Logic in Computer Science, Philadelphia*, June 1990.

[Gir87a] J.-Y. Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.

[Gir87b] J.-Y. Girard. Multiplicatives. *Rendiconti del Seminario Matematico dell' Universitá e Politecnico Torino, Special Issue on Logic and Computer Science*, pages 11–33, 1987.

[Gir89] J.-Y. Girard. Towards a geometry of interaction. In: Contemporary Math. 92, Amer. Math. Soc., 1989. 69-108.

[Gir90] J.-Y. Girard. La logique linéaire. *Pour La Science, Édition Francaise de Scientific American*, 150:74–85, April 1990.

[GJ79] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W.H. Freeman and Co., 1979.

[GLT89] J.-Y. Girard, Y. Lafont, and P. Taylor. *Proofs and Types.* Cambrige Tracts in Theoretical Computer Science, Cambridge University Press, 1989.

[GSS90] J.-Y. Girard, A. Scedrov, and P.J. Scott. Bounded linear logic: A modular approach to polynomial time computability. In *Proc. Math. Sci. Institute Workshop on Feasible Mathematics, Cornell University, June, 1988.* Birkhauser, 1990.

[HS86] J.R. Hindley and J.P. Seldin. *Introduction to Combinators and Lambda Calculus.* London Mathematical Society Student Texts, Cambridge University Press, 1986.

[HU79] J. Hopcroft and J. Ullman. *Introduction to Automata Theory, Languages and Computation.* Addison-Wesley Publishing Company, 1979.

[Kos82] S.R. Kosaraju. Decidability of reachability in vector addition systems. In *Proc. 14-th ACM Symp. on Theory of Computing*, pages 267–281, 1982.

[KW84] J. Ketonen and R. Weyhrauch. A decidable fragment of predicate calculus. *Theoretical Computer Science*, 32, 1984.

[Laf88] Y. Lafont. The linear abstract machine. *Theoretical Computer Science*, 59:157–180, 1988.

[Laf90] Y. Lafont. Interaction nets. In *Proc. 17-th ACM Symp. on Principles of Programming Languages, San Francisco*, pages 95–108, January 1990.

[Lam58] J. Lambek. The mathematics of sentence structure. *Amer. Math. Monthly*, 65:154–169, 1958.

[Lam61] J. Lambek. How to program an infinite abacus. *Canadian Math. Bulletin*, 4:295–302, 1961.

[Lip76] R. Lipton. The reachability problem is exponential-space hard. Technical Report 62, Department of Computer Science, Yale University, January 1976.

[LMSS90] P. Lincoln, J.C. Mitchell, A. Scedrov, and N. Shankar. Decision problems for propositional linear logic. In *Proc. 31st IEEE Symp. on Foundations of Computer Science*, pages 662–671, 1990.

[May81] E.W. Mayr. An algorithm for the general Petri net reachability problem. In *Proc. 13-th ACM Symp. on Theory of Computing, Milwaukee*, pages 238–246, 1981.

[McA84] K. McAloon. Petri nets and large sets. *Theoretical Computer Science*, 32:173–183, 1984.

[Mey66] R.K. Meyer. *Topics in Modal and Many-Valued Logic.* PhD thesis, University of Pittsburgh, 1966.

[Min61] M. Minsky. Recursive unsolvability of Post's problem of 'tag' and other topics in the theory of Turing machines. *Annals of Mathematics*, 74:3:437–455, 1961.

[MM82] E. Mayr and A. Meyer. The complexity of the word problems for commutative semigroups and polynomial ideals. *Advances in Mathematics*, 46:305–329, 1982.

[MOM89] N. Martí-Oliet and J. Meseguer. From Petri nets to linear logic. In: Springer LNCS 389, ed. by D.H. Pitt et al., 1989. 313-340.

[Pos47] E.L. Post. Recursive unsolvability of a problem of Thue. *Journal of Symbolic Logic*, 12:1–11, 1947.

[Sce90] A. Scedrov. A brief guide to linear logic. *Bulletin of the European Assoc. for Theoretical Computer Science*, 41:154–165, June 1990.

[Urq84] A. Urquhart. The undecidability of entailment and relevant implication. *Journal of Symbolic Logic*, 49:1059–1073, 1984.

[Yet90] D.N. Yetter. Quantales and (noncommutative) linear logic. *Journal of Symbolic Logic*, 55:41–64, 1990.

Appendix A

Cut Elimination

The cut-elimination theorem, in general, states that whatever can be proven in the full version of a logic may also be proven without the use of the cut rule. This theorem is fundamental to linear logic, and was proven by Girard shortly after the introduction of the logic by presenting a cut-elimination procedure for proof nets [Gir87a]. In our proof of undecidability, we make use of the syntax, or exact form of a cut-elimination procedure for the sequent calculus formulation of linear logic. Since Girard demonstrated the correspondence between proof nets and the sequent calculus presentation of linear logic, we could have relied on Girard's proof of cut-elimination. However, for the purposes of our undecidability proof, and other results, it is much more clear to present a cut-elimination procedure native to the sequent calculus.

The following demonstration of the cut-elimination theorem consists of a linear logic proof normalization procedure which slowly eliminates cuts from any linear logic proof. The procedure may greatly increase the size of the proof, although of course it will still be a proof of the same sequent. For technical reasons, we add a derived rule of inference, **Cut!**, which simplifies the proof of termination. We then give a set of reductions which apply to proofs which end in **Cut** or **Cut!**, and using these we eliminate all uses of **Cut** and **Cut!** from a proof.

The proof structure is very close to the well known proofs of cut-elimination in classical logic [GLT89], but is complicated by the extra information which must be preserved in a linear proof. The **Cut!** rule defined below is reminiscent of Gentzen's MIX rule [Gen69], and serves the same purpose, which is to package together inference rules. As in Gentzen's work, we add this extra rule, and then show that it (along with **Cut**) may be eliminated entirely from any proof. Thus we show that this new rule and **Cut** are redundant in linear logic.

Let us begin with some definitions. First, we define the following new rule of inference,

$$\textbf{Cut!} \quad \frac{\vdash \Sigma, (?A)^n \qquad \vdash \Delta, !A^\perp}{\vdash \Sigma, \Delta} \quad n \geq 1$$

$(?A)^n$ is meant to denote a multiset of formulas. For example, $(?A)^3 = ?A, ?A, ?A$. As stated in the side condition, the **Cut!** rule is only applicable when n is at least 1. This rule of inference is derivable; that is, it may be simulated by several applications of contraction (?**C**) on the left hypothesis and then one application of the standard **Cut** rule. The original **Cut** rule and **Cut!** coincide when $n = 1$. Adding this extra derived rule of inference simplifies the termination argument substantially by packaging together some number of contractions with the cut that eliminates the contracted formula. This package is only opened when the contracted formulas are actually used with the application of the ?**D** rule, thrown away by the ?**W** rule, or split into two packages by the $\otimes$ and **Cut**$*$ rules. We will use the symbol "**Cut**$*$" as a general term for the original **Cut** rule and the new **Cut!** rule ambiguously.

We will call a formula which appears in a hypothesis of an application of **Cut** or **Cut!**, but which does not occur in the conclusion a *cut-formula*. In the list of linear logic rules in Appendix B the cut-formulas in the **Cut** rule are the formulas named A and $A^\perp$, and in the **Cut!** rule above, the cut-formulas are $?A$ and $!A^\perp$.

We also define the *degree* of a **Cut** or **Cut!** to be the number of symbols in its cut-formulas. For concreteness, we define here what is meant by number of symbols. We will consider each propositional symbol p_i to be a single symbol. We also consider the negation of each propositional symbol $p_i{}^\perp$ to be a single symbol. Finally, we count each connective and constant, $\otimes$, $\parr$, $\oplus$, $\&$, $?$, $!$, 1, $\perp$, 0, $\top$, as a single symbol, but do not count parentheses. It is important to note that negation is defined, and therefore is not a connective. This method of accounting has the pleasant property that any linear logic formula A and its negation $A^\perp$ have exactly the same number of symbols. (One may prove this by simple induction on the structure of the formula A). Thus it does not matter which cut-formula we count when determining the degree of a cut. We also define the *degree* of a proof to be the maximum degree of any cut in the proof, or zero if there are no cuts.

The *principal formula* of an application of an inference rule is usually defined to be any formula which is introduced by that rule. For example, the formula $(A \otimes B)$ is the principal formula of the $\otimes$ rule, since that is the formula which is introduced by that rule. We follow the standard convention of considering the contracted formula in an application of $?C$ principal, even though it is not introduced by the rule. For convenience, we extend the notion of principal formula in the following nonstandard ways. We will consider any formula beginning with ? appearing in the conclusion of the **!S**, $\otimes$, **Cut**, or **Cut!** rules to be principal. By this definition all formulas in the conclusion of **!S** are principal, and the only rule in which a formula beginning with ! may be principal is **!S**. This definition of principal formula simplifies the structure of the following proof somewhat.

Operationally, the cut-elimination procedure defined below first finds one of the "highest" cuts of maximal degree in the proof. That is, an application of **Cut**$*$ (**Cut** or **Cut**!) for which all applications of **Cut**$*$ in the derivation of either hypothesis is of smaller degree. Then a reduction is applied to that occurrence of **Cut**$*$, which simplifies or eliminates it, although it may replicate some other portions of the original proof. We iterate this procedure to remove all cuts of some degree, and then iterate the entire procedure to eliminate all cuts. In this way, any linear logic proof may be normalized into one without any uses of the **Cut** or **Cut**! rules, at the possible expense of an (worse than) exponential blowup in the size of the resulting proof tree.

Technically, we begin with a lemma which constitutes the heart of the proof of cut-elimination. Although the proof of this lemma is rather lengthy, the reasoning is straightforward, and the remainder of the proof of cut-elimination is quite simple.

Lemma A.1 (Reduce One Cut) *Given a proof of the sequent* $\vdash \Gamma$ *in linear logic which ends in an application of* **Cut**$*$ *of degree* $d > 0$*, and where the degree of the proofs of both hypothesis is less than* d*, we may construct a proof of* $\vdash \Gamma$ *in linear logic of degree less than* d*.*

Proof. By induction on the number of proof rules applied in the derivation of $\vdash \Gamma$.

Given a derivation which ends in a **Cut**$*$, we perform case analysis on the rules which were applied immediately above the **Cut**$*$. One of the following cases must apply to any such derivation:

1. The cut-formula is *not* principal in one or both hypotheses.

2. The cut-formula *is* principal in both hypotheses.

In each case we will provide a reduction, which may eliminate the cut entirely, or replace it with one or two smaller cuts. Since this is a proof by induction on the size of a derivation, one may view this proof as a procedure which pushes applications of **Cut**$*$ of large degree up a derivation. Informally, this procedure pushes applications of **Cut**$*$ up through proof rules where the cut-formula is non-principal, until the critical point is reached where the cut-formula is principal in both hypotheses. In Girard's proof of cut-elimination for linear logic using proof nets, the non-principal cases are circumvented by following proof links. In both approaches, however, the principal cases require significant detailed analysis.

A.1 Cut of non-principal formulas

If the derivation of a hypothesis ends in a rule yielding a non-principal cut-formula, then the rule must be one of the following: $\otimes$, $\parr$, $\oplus$, $\&$, ?**W**, ?**C**, ?**D**, $\bot$, $\top$, or

Cut$*$. The rules **I**, **!S**, and 1 are absent since those rules have no non-principal formulas in their conclusions. The later analysis of principal formula cuts considers these three cases.

A.1.1 $\otimes$

If the last rule applied in one hypothesis is $\otimes$, the cut-formula is not the main formula introduced by that application of $\otimes$, and the cut-formula does not begin with ?, then we may propagate the **Cut**$*$ upward, through the application of $\otimes$:

$$\dfrac{\dfrac{\begin{array}{c}\vdots\\ \vdash \Sigma, A\end{array} \quad \begin{array}{c}\vdots\\ \vdash B, \Delta, C\end{array}}{\vdash \Sigma, (A \otimes B), \Delta, C}\otimes \quad \begin{array}{c}\vdots\\ \vdash \Gamma, C^{\perp}\end{array}}{\vdash \Sigma, (A \otimes B), \Delta, \Gamma}\mathbf{Cut} \quad \Longrightarrow \quad \dfrac{\begin{array}{c}\vdots\\ \vdash \Sigma, A\end{array} \quad \dfrac{\begin{array}{c}\vdots\\ \vdash B, \Delta, C\end{array} \quad \begin{array}{c}\vdots\\ \vdash \Gamma, C^{\perp}\end{array}}{\vdash B, \Delta, \Gamma}\mathbf{Cut}}{\vdash \Sigma, (A \otimes B), \Delta, \Gamma}\otimes$$

For the rules such as $\otimes$ with two hypotheses, we give the reduction for the case where the non-principal cut-formula appears in the right hand hypothesis of the $\otimes$ rule, and appears in one specific position in that sequent. The symmetric case of the cut-formula appearing in the left hand hypothesis is very similar, and is always omitted. Since some notion of exchange is built-in to the system, sequents are considered multisets. Thus the exact position of formulas in sequents is unimportant. (Note that in noncommutative linear logic the relative position becomes vitally important.)

The proof ending in **Cut** after this transformation is smaller than the original proof, since the entire proof of $\vdash \Sigma, A$, and the last application of $\otimes$ are no longer above the **Cut**. Thus by induction on the size of proofs, we can construct the desired proof of degree less than d.

Note that the **Cut!** rule only applies to formulas which begin with ?, and thus this reduction, which is only used if the cut-formula does not begin with ?, applies only to **Cut** and not to **Cut!**. Thus, we have disambiguated this case, and write only of **Cut**, where we present tranformations later in terms of **Cut**$*$, in order to cover both possibilities simultaneously. The reduction given later (in Section A.2.9) handles the case of **Cut!**.

A.1.2 $\parr$

If the last rule applied in one hypothesis is $\parr$, and the cut-formula is not the main formula introduced by that application of $\parr$, then we may propagate the **Cut**$*$

upward, through the application of $\parr$:

$$\dfrac{\dfrac{\begin{array}{c}\vdots\\ \vdash \Sigma, A, B, C\end{array}}{\vdash \Sigma, (A \parr B), C}\parr \qquad \begin{array}{c}\vdots\\ \vdash \Gamma, C^{\perp}\end{array}}{\vdash \Sigma, (A \parr B), \Gamma}\mathbf{Cut}* \quad \Longrightarrow \quad \dfrac{\dfrac{\begin{array}{c}\vdots\\ \vdash \Sigma, A, B, C\end{array} \qquad \begin{array}{c}\vdots\\ \vdash \Gamma, C^{\perp}\end{array}}{\vdash \Sigma, A, B, \Gamma}\mathbf{Cut}*}{\vdash \Sigma, (A \parr B), \Gamma}\parr$$

Again, the proof above the **Cut**$*$ is smaller after this transformation, and thus by induction we have our result.

A.1.3 $\oplus$

Applications of **Cut**$*$ involving the two symmetric $\oplus$ rules (where the cut-formula is not principle, that is, not introduced by this application of $\oplus$) may be eliminated in similar ways:

$$\dfrac{\dfrac{\begin{array}{c}\vdots\\ \vdash \Sigma, A, C\end{array}}{\vdash \Sigma, (A \oplus B), C}\oplus \qquad \begin{array}{c}\vdots\\ \vdash \Gamma, C^{\perp}\end{array}}{\vdash \Sigma, (A \oplus B), \Gamma}\mathbf{Cut}* \quad \Longrightarrow \quad \dfrac{\dfrac{\begin{array}{c}\vdots\\ \vdash \Sigma, A, C\end{array} \qquad \begin{array}{c}\vdots\\ \vdash \Gamma, C^{\perp}\end{array}}{\vdash \Sigma, A, \Gamma}\mathbf{Cut}*}{\vdash \Sigma, (A \oplus B), \Gamma}\oplus$$

The second case of this rule is the same except the conclusion would contain the formula $(B \oplus A)$, instead of the formula $(A \oplus B)$ seen above.

A.1.4 &

It is the elimination of this type of cut (among others) which may lead to an exponential blowup in the size of cut-free proofs. The only other cut-elimination steps which may lead to a proof expansion are those involving **!S**.

$$\dfrac{\dfrac{\begin{array}{c}\vdots\\ \vdash \Sigma, A, C\end{array} \quad \begin{array}{c}\vdots\\ \vdash \Sigma, B, C\end{array}}{\vdash \Sigma, (A\&B), C}\& \qquad \begin{array}{c}\vdots\\ \vdash \Gamma, C^{\perp}\end{array}}{\vdash \Sigma, (A\&B), \Gamma}\mathbf{Cut}*$$

$$\Downarrow$$

$$\dfrac{\dfrac{\begin{array}{c}\vdots\\ \vdash \Sigma, A, C\end{array} \quad \begin{array}{c}\vdots\\ \vdash \Gamma, C^{\perp}\end{array}}{\vdash \Sigma, A, \Gamma}\mathbf{Cut}* \quad \dfrac{\begin{array}{c}\vdots\\ \vdash \Sigma, B, C\end{array} \quad \begin{array}{c}\vdots\\ \vdash \Gamma, C^{\perp}\end{array}}{\vdash \Sigma, B, \Gamma}\mathbf{Cut}*}{\vdash \Sigma, (A\&B), \Gamma}\&$$

The increase in proof size comes from replicating the entire proof tree above $\vdash \Gamma, C^{\perp}$. Note that even though there are now two cuts instead of one, we may assume that both may be reduced in degree to less than d by induction on the size of the derivations. That is, there are fewer proof rules applied above each **Cut**$*$ than there were above the single application of **Cut**$*$ originally.

A.1.5 ?W

For this and the remaining cases, we omit discussion and simply indicate the reduction:

$$\dfrac{\dfrac{\begin{array}{c}\vdots\\ \vdash \Sigma, A\end{array}}{\vdash \Sigma, A, ?B}\,?\mathbf{W} \qquad \begin{array}{c}\vdots\\ \vdash \Gamma, A^{\perp}\end{array}}{\vdash \Sigma, \Gamma, ?B}\,\mathbf{Cut}* \quad \Longrightarrow \quad \dfrac{\dfrac{\begin{array}{c}\vdots\\ \vdash \Sigma, A\end{array} \qquad \begin{array}{c}\vdots\\ \vdash \Gamma, A^{\perp}\end{array}}{\vdash \Sigma, \Gamma}\,\mathbf{Cut}*}{\vdash \Sigma, \Gamma, ?B}\,?\mathbf{W}$$

A.1.6 ?C

$$\dfrac{\dfrac{\begin{array}{c}\vdots\\ \vdash \Sigma, A, ?B, ?B\end{array}}{\vdash \Sigma, A, ?B}\,?\mathbf{C} \qquad \begin{array}{c}\vdots\\ \vdash \Gamma, A^{\perp}\end{array}}{\vdash \Sigma, \Gamma, ?B}\,\mathbf{Cut}* \quad \Longrightarrow \quad \dfrac{\dfrac{\begin{array}{c}\vdots\\ \vdash \Sigma, A, ?B, ?B\end{array} \qquad \begin{array}{c}\vdots\\ \vdash \Gamma, A^{\perp}\end{array}}{\vdash \Sigma, \Gamma, ?B, ?B}\,\mathbf{Cut}*}{\vdash \Sigma, \Gamma, ?B}\,?\mathbf{C}$$

A.1.7 ?D

$$\dfrac{\dfrac{\begin{array}{c}\vdots\\ \vdash \Sigma, A, B\end{array}}{\vdash \Sigma, A, ?B}\,?\mathbf{D} \qquad \begin{array}{c}\vdots\\ \vdash \Gamma, A^{\perp}\end{array}}{\vdash \Sigma, \Gamma, ?B}\,\mathbf{Cut}* \quad \Longrightarrow \quad \dfrac{\dfrac{\begin{array}{c}\vdots\\ \vdash \Sigma, A, B\end{array} \qquad \begin{array}{c}\vdots\\ \vdash \Gamma, A^{\perp}\end{array}}{\vdash \Sigma, \Gamma, B}\,\mathbf{Cut}*}{\vdash \Sigma, \Gamma, ?B}\,?\mathbf{D}$$

A.1.8 $\perp$

$$\dfrac{\dfrac{\begin{array}{c}\vdots\\ \vdash \Sigma, A\end{array}}{\vdash \Sigma, A, \perp}\,\perp \qquad \begin{array}{c}\vdots\\ \vdash \Gamma, A^{\perp}\end{array}}{\vdash \Sigma, \Gamma, \perp}\,\mathbf{Cut}* \quad \Longrightarrow \quad \dfrac{\dfrac{\begin{array}{c}\vdots\\ \vdash \Sigma, A\end{array} \qquad \begin{array}{c}\vdots\\ \vdash \Gamma, A^{\perp}\end{array}}{\vdash \Sigma, \Gamma}\,\mathbf{Cut}*}{\vdash \Sigma, \Gamma, \perp}\,\perp$$

A.1.9 $\top$

$$\dfrac{\dfrac{}{\vdash \Sigma, A, \top}\top \quad \begin{array}{c}\vdots\\ \vdash \Gamma, A^{\perp}\end{array}}{\vdash \Sigma, \Gamma, \top}\mathbf{Cut}* \Longrightarrow \dfrac{}{\vdash \Sigma, \Gamma, \top}\top$$

A.1.10 Cut

If the proof of one hypothesis ends in **Cut**$*$, then we know that it has degree less than d. If the cut-formula of the lower degree d application of **Cut**$*$ begins with ?, then it is considered principal (by definition) in the upper application of **Cut**$*$, and will be handled in Section A.2.8. Otherwise, we know the formula does not begin with ?, and thus the lower **Cut**$*$ must actually be **Cut**.

$$\dfrac{\dfrac{\begin{array}{c}\vdots\\ \vdash \Sigma, A, C\end{array} \quad \begin{array}{c}\vdots\\ \vdash \Gamma, C^{\perp}\end{array}}{\vdash \Sigma, A, \Gamma}\mathbf{Cut}* \quad \begin{array}{c}\vdots\\ \vdash \Delta, A^{\perp}\end{array}}{\vdash \Sigma, \Delta, \Gamma}\mathbf{Cut} \Longrightarrow \dfrac{\dfrac{\begin{array}{c}\vdots\\ \vdash \Sigma, A, C\end{array} \quad \begin{array}{c}\vdots\\ \vdash \Delta, A^{\perp}\end{array}}{\vdash \Sigma, \Delta, C}\mathbf{Cut} \quad \begin{array}{c}\vdots\\ \vdash \Gamma, C^{\perp}\end{array}}{\vdash \Sigma, \Delta, \Gamma}\mathbf{Cut}*$$

Here we know that the number of symbols in the formula A is d, and the number of symbols in the formula C is less than d. Thus by induction we know that we can construct a proof of degree less than d of $\vdash \Sigma, \Delta, C$, and from that we can construct our desired proof of $\vdash \Sigma, \Delta, \Gamma$.

A.2 Cut of principal formulas

If the proof of each hypothesis ends in a rule with the cut formula as its principal formula, then the two last rules above the cut must be one of these combinations: **I** versus any, $\otimes$ versus $\parr$, $\oplus$ versus &, ?**W** versus !**S**, ?**C** versus !**S**, ?**D** versus !**S**, !**S** versus !**S**, **Cut**$*$ versus !**S**, $\otimes$ versus !**S**, or $\perp$ versus 1. Note that since there is no introduction rule for 0, the $\top$ rule cannot participate in a cut of principal formulas. Since all formulas in the conclusion of !**S** are considered principal, the analysis of !**S** at this stage of the proof is rather complex.

In many of these cases, we know that the **Cut**! rule is inapplicable, since the cut-formula has just been introduced, and it does not begin with a ?. When we know this, we will disambiguate the reduction, and show the applications of **Cut** and **Cut**! separately.

A.2.1 I versus any

If the last rule applied in either hypothesis is **I** (identity), then regardless of the rule applied in the other hypothesis we may remove the cut, and the application of identity:

$$\dfrac{\dfrac{}{\vdash p_i, p_i^{\perp}}\mathbf{I} \qquad \begin{array}{c}\vdots\\ \vdash p_i, \Gamma\end{array}}{\vdash p_i, \Gamma}\mathbf{Cut} \Longrightarrow \begin{array}{c}\vdots\\ \vdash p_i, \Gamma\end{array}$$

Note that the identity axiom only applies to atomic propositions, and thus we know that **Cut!** is inapplicable.

A.2.2 $\otimes$ versus $\parr$

$$\dfrac{\dfrac{\begin{array}{c}\vdots\\ \vdash \Sigma, A\end{array} \qquad \begin{array}{c}\vdots\\ \vdash B, \Delta\end{array}}{\vdash \Sigma, (A \otimes B), \Delta}\otimes \qquad \dfrac{\begin{array}{c}\vdots\\ \vdash \Gamma, B^{\perp}, A^{\perp}\end{array}}{\vdash \Gamma, (B^{\perp} \parr A^{\perp})}\parr}{\vdash \Sigma, \Gamma, \Delta}\mathbf{Cut}$$

$$\Downarrow$$

$$\dfrac{\begin{array}{c}\vdots\\ \vdash \Sigma, A\end{array} \qquad \dfrac{\begin{array}{c}\vdots\\ \vdash B, \Delta\end{array} \qquad \begin{array}{c}\vdots\\ \vdash \Gamma, B^{\perp}, A^{\perp}\end{array}}{\vdash \Gamma, \Delta, A^{\perp}}\mathbf{Cut}}{\vdash \Gamma, \Delta, \Sigma}\mathbf{Cut}$$

In this case, as in most of the principal formula cut-elimination steps, we need not appeal to the induction hypothesis of this lemma. We have eliminated the **Cut** of degree d, and replaced it with two applications of **Cut** of degree smaller than d.

A.2.3 & versus $\oplus$

$$\dfrac{\dfrac{\begin{array}{c}\vdots\\ \vdash \Sigma, A\end{array} \qquad \begin{array}{c}\vdots\\ \vdash \Sigma, B\end{array}}{\vdash \Sigma, (A \& B)}\& \qquad \dfrac{\begin{array}{c}\vdots\\ \vdash \Gamma, A^{\perp}\end{array}}{\vdash \Gamma, (A^{\perp} \oplus B^{\perp})}\oplus}{\vdash \Sigma, \Gamma}\mathbf{Cut} \Longrightarrow \dfrac{\begin{array}{c}\vdots\\ \vdash \Sigma, A\end{array} \qquad \begin{array}{c}\vdots\\ \vdash \Gamma, A^{\perp}\end{array}}{\vdash \Sigma, \Gamma}\mathbf{Cut}$$

The symmetric case of $\oplus$ is similar. Again, we need not appeal to the induction hypothesis, and the cut-formula does not begin with ?, and thus we know that **Cut!** does not apply.

A.2.4 ?W versus !S

For this and subsequent cases involving !S, 'packaging' is a useful analogy. We build packages containing a number of contractions and a single **Cut**! when we reduce principal cases involving ?**C** versus !**S**. We shrink the package in cases of ?**W** versus !**S**, and we actually use the contents of the package as cases of ?**D** versus !**S**. We let packages pass by each other at cases of !**S** versus !**S**, and at cases of **Cut**! versus !**S** and of $\otimes$ versus !**S** we break one package into two.

For this case, ?**W** versus !**S**, there are two possibilities, depending on whether the cut in question eliminates more than one occurrence of the cut-formula from the weakened sequent. Informally, the possibilities turn on whether there is only one thing in the package. If so, we don't need the package. If there are more things in the package, we shrink the package.

In the first possibility, the cut eliminates the one occurrence of the cut-formula introduced by the ?**W** rule, and thus this application of cut may be eliminated entirely:

$$\dfrac{\dfrac{\begin{array}{c}\vdots\\ \vdash \Sigma\end{array}}{\vdash \Sigma, ?A}\,?\mathbf{W} \quad \dfrac{\begin{array}{c}\vdots\\ \vdash ?\Gamma, A^{\perp}\end{array}}{\vdash ?\Gamma, !A^{\perp}}\,!\mathbf{S}}{\vdash \Sigma, ?\Gamma}\,\mathbf{Cut}* \quad \Longrightarrow \quad \dfrac{\begin{array}{c}\vdots\\ \dfrac{\vdash \Sigma}{}\,?\mathbf{W}\\ \vdots\end{array}}{\vdash \Sigma, ?\Gamma}\,?\mathbf{W}$$

However, the second possibility, where the **Cut**$*$ is actually a **Cut**! and eliminates more than one occurrence of the cut-formula from the weakened sequent, we perform the following reduction:

$$\dfrac{\dfrac{\begin{array}{c}\vdots\\ \vdash \Sigma, (?A)^{n-1}\end{array}}{\vdash \Sigma, (?A)^{n}}\,?\mathbf{W} \quad \dfrac{\begin{array}{c}\vdots\\ \vdash ?\Gamma, A^{\perp}\end{array}}{\vdash ?\Gamma, !A^{\perp}}\,!\mathbf{S}}{\vdash \Sigma, ?\Gamma}\,\mathbf{Cut!} \quad \Longrightarrow \quad \dfrac{\begin{array}{c}\vdots\\ \vdash \Sigma, (?A)^{n-1}\end{array} \quad \begin{array}{c}\vdots\\ \dfrac{\vdash ?\Gamma, A^{\perp}}{\vdash ?\Gamma, !A^{\perp}}\,!\mathbf{S}\end{array}}{\vdash \Sigma, ?\Gamma}\,\mathbf{Cut!}$$

In the first possibility we have our result immediately, since the **Cut**$*$ has been eliminated. In the second possibility, we appeal to the induction hypothesis.

A.2.5 ?C versus !S

In this case we make critical use of the **Cut**! rule. Without this extra rule of inference this reduction is especially difficult to formulate correctly, and the induction required

is complicated.

$$\dfrac{\dfrac{\begin{array}{c}\vdots\\ \vdash \Sigma, ?A, ?A\end{array}}{\vdash \Sigma, ?A}{\scriptstyle ?\mathbf{C}} \quad \dfrac{\begin{array}{c}\vdots\\ \vdash ?\Gamma, A^{\perp}\end{array}}{\vdash ?\Gamma, !A^{\perp}}{\scriptstyle !\mathbf{S}}}{\vdash \Sigma, ?\Gamma}{\scriptstyle \mathbf{Cut}*} \quad \Longrightarrow \quad \dfrac{\begin{array}{c}\vdots\\ \vdash \Sigma, ?A, ?A\end{array} \quad \dfrac{\begin{array}{c}\vdots\\ \vdash ?\Gamma, A^{\perp}\end{array}}{\vdash ?\Gamma, !A^{\perp}}{\scriptstyle !\mathbf{S}}}{\vdash \Sigma, ?\Gamma}{\scriptstyle \mathbf{Cut}!}$$

Here we know that the cut-formula begins with a ?, and thus **Cut**! may apply to it. We thus produce a **Cut**! regardless of whether the original **Cut**$*$ was a **Cut** or a **Cut**!.

A.2.6 ?D versus !S

As for the previous ?**W** versus !**S** case, here we have two cases, depending on whether the **Cut**$*$ in question eliminates more than one occurrence of the cut-formula from the derelicted sequent. Again, informally, the two cases turn on the size of the package. If there is only one thing in the package, we simple make use of it, and throw away the wrapping. If there are more thing in the package, we take one out, and move the smaller package along its way.

In the first case, the cut eliminates the one occurrence of the cut-formula introduced by the ?**D** rule, and thus the following reduction applies:

$$\dfrac{\dfrac{\begin{array}{c}\vdots\\ \vdash \Sigma, A\end{array}}{\vdash \Sigma, ?A}{\scriptstyle ?\mathbf{D}} \quad \dfrac{\begin{array}{c}\vdots\\ \vdash ?\Gamma, A^{\perp}\end{array}}{\vdash ?\Gamma, !A^{\perp}}{\scriptstyle !\mathbf{S}}}{\vdash \Sigma, ?\Gamma}{\scriptstyle \mathbf{Cut}*}$$

$$\Downarrow$$

$$\dfrac{\begin{array}{c}\vdots\\ \vdash \Sigma, A\end{array} \quad \begin{array}{c}\vdots\\ \vdash ?\Gamma, A^{\perp}\end{array}}{\vdash \Sigma, ?\Gamma}{\scriptstyle \mathbf{Cut}}$$

However, in the second case, where the cut is actually a **Cut**! and eliminates more than one occurrence of the cut-formula from the derelicted sequent, we perform the

following reduction:

$$\dfrac{\dfrac{\vdots}{\vdash \Sigma, ?A^{n-1}, A}\,{}_{?\mathbf{D}} \qquad \dfrac{\vdots}{\vdash ?\Gamma, A^{\perp}}}{\vdash \Sigma, ?\Gamma}\,{}_{\mathbf{Cut!}} \quad \Longrightarrow \quad \dfrac{\dfrac{\dfrac{\dfrac{\vdots}{\vdash \Sigma, ?A^{n-1}, A} \qquad \dfrac{\dfrac{\vdots}{\vdash ?\Gamma, A^{\perp}}}{\vdash ?\Gamma, !A^{\perp}}\,{}_{!\mathbf{S}}}{\vdash \Sigma, ?\Gamma, A}\,{}_{\mathbf{Cut!}} \qquad \dfrac{\vdots}{\vdash ?\Gamma, A^{\perp}}}{\vdash \Sigma, ?\Gamma, ?\Gamma}\,{}_{\mathbf{Cut}}}{\dfrac{\vdots}{\vdash \Sigma, ?\Gamma}\,{}_{?\mathbf{C}}}\,{}_{?\mathbf{C}}$$

Note that the second case requires the duplication of the proof above the application of **!S**. Since A has fewer symbols than $?A$, the lower **Cut** in the second case is of degree smaller than d. By induction, we may assume that the upper application of **Cut!** is reducible in degree.

A.2.7 !S versus !S

$$\dfrac{\dfrac{\vdots}{\vdash ?\Sigma, ?A, B}}{\vdash ?\Sigma, ?A, !B}\,{}_{!\mathbf{S}} \qquad \dfrac{\vdots}{\vdash ?\Gamma, A^{\perp}}}{\vdash ?\Gamma, !A^{\perp}}\,{}_{!\mathbf{S}}}{\vdash ?\Sigma, ?\Gamma, !B}\,{}_{\mathbf{Cut}*} \quad \Longrightarrow \quad \dfrac{\dfrac{\vdots}{\vdash ?\Sigma, ?A, B} \qquad \dfrac{\dfrac{\vdots}{\vdash ?\Gamma, A^{\perp}}}{\vdash ?\Gamma, !A^{\perp}}\,{}_{!\mathbf{S}}}{\vdash ?\Sigma, ?\Gamma, B}\,{}_{\mathbf{Cut}*}}{\vdash ?\Sigma, ?\Gamma, !B}\,{}_{!\mathbf{S}}$$

Here we appeal to the induction hypothesis to produce a proof degree less than d of $\vdash ?\Sigma, ?\Gamma, B$, and then construct the desired proof from that.

A.2.8 Cut∗ versus !S

There are two possibilities here, which correspond to whether it is necessary to split a package into two pieces. The case where the package needs to be split is one of the most tricky aspects of the entire cut-elimination procedure.

If the lower application of **Cut∗** is applied to formulas which may all be found in one hypothesis of the upper application of **Cut∗**, then we apply the same reduction

as in the non-principal **Cut** case (Section A.1.10):

$$\dfrac{\dfrac{\begin{array}{c}\vdots\\ \vdash \Sigma, A\end{array} \quad \begin{array}{c}\vdots\\ \vdash \Delta, (?C)^n, A^\perp\end{array}}{\vdash \Sigma, \Delta, (?C)^n}\mathbf{Cut}* \quad \dfrac{\begin{array}{c}\vdots\\ \vdash ?\Gamma, C^\perp\end{array}}{\vdash ?\Gamma, !C^\perp}\mathbf{!S}}{\vdash \Sigma, \Delta, ?\Gamma}\mathbf{Cut}*$$

$$\Downarrow$$

$$\dfrac{\begin{array}{c}\vdots\\ \vdash \Sigma, A\end{array} \quad \dfrac{\begin{array}{c}\vdots\\ \vdash \Delta, (?C)^n, A^\perp\end{array} \quad \dfrac{\begin{array}{c}\vdots\\ \vdash ?\Gamma, C^\perp\end{array}}{\vdash ?\Gamma, !C^\perp}\mathbf{!S}}{\vdash \Delta, ?\Gamma, A^\perp}\mathbf{Cut}*}{\vdash \Sigma, \Delta, ?\Gamma}\mathbf{Cut}*$$

In the more complex case, when the cut-formulas descend from both hypotheses of the upper **Cut**$*$,we use the following reduction:

$$\dfrac{\dfrac{\begin{array}{c}\vdots\\ \vdash \Sigma, ?C^m, A\end{array} \quad \begin{array}{c}\vdots\\ \vdash \Delta, ?C^n, A^\perp\end{array}}{\vdash \Sigma, \Delta, ?C^{n+m}}\mathbf{Cut}* \quad \dfrac{\begin{array}{c}\vdots\\ \vdash ?\Gamma, C^\perp\end{array}}{\vdash ?\Gamma, !C^\perp}\mathbf{!S}}{\vdash \Sigma, \Delta, ?\Gamma}\mathbf{Cut!}$$

$$\Downarrow$$

$$\dfrac{\dfrac{\dfrac{\begin{array}{c}\vdots\\ \vdash \Sigma, ?C^m, A\end{array} \quad \dfrac{\begin{array}{c}\vdots\\ \vdash ?\Gamma, C^\perp\end{array}}{\vdash ?\Gamma, !C^\perp}\mathbf{!S}}{\vdash \Sigma, ?\Gamma, A}\mathbf{Cut!} \quad \dfrac{\begin{array}{c}\vdots\\ \vdash \Delta, ?C^n, A^\perp\end{array} \quad \dfrac{\begin{array}{c}\vdots\\ \vdash ?\Gamma, C^\perp\end{array}}{\vdash ?\Gamma, !C^\perp}\mathbf{!S}}{\vdash \Delta, ?\Gamma, A^\perp}\mathbf{Cut!}}{\vdash \Sigma, ?\Gamma, \Delta, ?\Gamma}\mathbf{Cut}*}{\begin{array}{c}\vdots\\ \vdash \Sigma, \Delta, ?\Gamma,\end{array}}\mathbf{?C}$$

A.2.9 $\otimes$ versus !S

There are two possibilities here, which correspond to whether it is necessary to split a package into two pieces. The case where the package needs to be split is again one of the most tricky aspects of the entire cut-elimination procedure.

If **Cut**$*$ is applied to formulas which may all be found in one hypothesis of $\otimes$,

then we apply the same reduction as in the non-principal $\otimes$ case (Section A.1.1):

$$\dfrac{\dfrac{\begin{array}{c}\vdots\\ \vdash \Sigma, A\end{array} \quad \begin{array}{c}\vdots\\ \vdash B, \Delta, (?C)^n\end{array}}{\vdash \Sigma, (A \otimes B), \Delta, (?C)^n}\otimes \quad \dfrac{\begin{array}{c}\vdots\\ \vdash ?\Gamma, C^\perp\end{array}}{\vdash ?\Gamma, !C^\perp}\mathbf{!S}}{\vdash \Sigma, (A \otimes B), \Delta, ?\Gamma}\mathbf{Cut*}$$

$$\Downarrow$$

$$\dfrac{\begin{array}{c}\vdots\\ \vdash \Sigma, A\end{array} \quad \dfrac{\begin{array}{c}\vdots\\ \vdash B, \Delta, (?C)^n\end{array} \quad \dfrac{\begin{array}{c}\vdots\\ \vdash ?\Gamma, C^\perp\end{array}}{\vdash ?\Gamma, !C^\perp}\mathbf{!S}}{\vdash B, \Delta, ?\Gamma}\mathbf{Cut*}}{\vdash \Sigma, (A \otimes B), \Delta, ?\Gamma}\otimes$$

In the more complex case, when the cut-formulas descend from both hypotheses of $\otimes$, we use the following reduction to push the cut above the $\otimes$ rule.

$$\dfrac{\dfrac{\begin{array}{c}\vdots\\ \vdash \Sigma, ?C^m, A\end{array} \quad \begin{array}{c}\vdots\\ \vdash B, \Delta, ?C^n\end{array}}{\vdash \Sigma, (A \otimes B), \Delta, ?C^{n+m}}\otimes \quad \dfrac{\begin{array}{c}\vdots\\ \vdash ?\Gamma, C^\perp\end{array}}{\vdash ?\Gamma, !C^\perp}\mathbf{!S}}{\vdash \Sigma, (A \otimes B), \Delta, ?\Gamma}\mathbf{Cut!}$$

$$\Downarrow$$

$$\dfrac{\dfrac{\dfrac{\begin{array}{c}\vdots\\ \vdash \Sigma, ?C^m, A\end{array} \quad \dfrac{\begin{array}{c}\vdots\\ \vdash ?\Gamma, C^\perp\end{array}}{\vdash ?\Gamma, !C^\perp}\mathbf{!S}}{\vdash \Sigma, ?\Gamma, A}\mathbf{Cut!} \quad \dfrac{\begin{array}{c}\vdots\\ \vdash B, \Delta, ?C^n\end{array} \quad \dfrac{\begin{array}{c}\vdots\\ \vdash ?\Gamma, C^\perp\end{array}}{\vdash ?\Gamma, !C^\perp}\mathbf{!S}}{\vdash B, \Delta, ?\Gamma}\mathbf{Cut!}}{\vdash \Sigma, ?\Gamma, (A \otimes B), \Delta, ?\Gamma}\otimes}{\begin{array}{c}\vdots\\ \vdash \Sigma, (A \otimes B), \Delta, ?\Gamma\end{array}}\mathbf{?C}}{\vdash \Sigma, (A \otimes B), \Delta, ?\Gamma}\mathbf{?C}$$

A.2.10 $\perp$ versus 1

$$\dfrac{\dfrac{}{\vdash 1}\mathbf{1} \quad \dfrac{\begin{array}{c}\vdots\\ \vdash \Gamma\end{array}}{\vdash \perp, \Gamma}\perp}{\vdash \Gamma}\mathbf{Cut} \quad \Longrightarrow \quad \begin{array}{c}\vdots\\ \vdash \Gamma\end{array}$$

Again, we know that the **Cut**$*$ involved here is **Cut**, since the formula 1 was just introduced.

This exhausts all the cases. ∎

Thus, we have a procedure which given a proof which ends in **Cut**$*$ of degree d, and which has no applications of **Cut**$*$ in the proof of either hypothesis of degree greater than or equal to d, produces a proof of degree less than d.

Lemma A.2 (Lower-Degree-Cuts) *If a sequent is provable in linear logic with a proof of degree $d > 0$, then it is provable in linear logic with a proof of degree less than d.*

Proof. By induction on the height of the derivation tree of the conclusion, we show that given any proof of degree d of $\vdash \Gamma$ in propositional linear logic, we may find a (possibly much larger) proof of $\vdash \Gamma$ in linear logic of degree less than d.

We examine the proof of $\vdash \Gamma$. Since the degree of this proof is greater than zero, there must be some **Cut**$*$ in the proof. If the last rule is not **Cut**$*$, then by induction we may form proofs of its hypotheses of degree less than d. Applying the same rule to the resulting reduced degree hypotheses produces the desired proof of degree less than d.

In the case that the last rule is **Cut**$*$, we have the following situation for some Σ and Δ which together (in multiset union) make up Γ:

$$\dfrac{\begin{array}{c}\vdots\\ \vdash \Sigma, A\end{array} \qquad \begin{array}{c}\vdots\\ \vdash \Delta, A^{\perp}\end{array}}{\vdash \Gamma}\,\mathbf{Cut}* \qquad \text{where } \Sigma \cup \Delta = \Gamma$$

By induction, we can produce proofs of $\vdash \Sigma, A$ and $\vdash \Delta, A^{\perp}$ of degree less than d. By a single application of Lemma A.1 to the resulting proof constructed from the modified hypotheses, we obtain a proof of $\vdash \Gamma$ of degree less than d. ∎

Theorem A.3 (Cut-Elimination) *If a sequent is provable in linear logic, then it is provable in linear logic without using the* **Cut** *rule.*

Proof. By induction on the degree of the assumed proof. We may apply Lemma A.2 at each inductive step, and at the base case the degree of the proof is zero, so therefore by definition of proof degree there are no cuts, and we have our desired cut-free proof. ∎

Note that the proof can explode hyperexponentially in size during the cut-elimination process.

A.3 Subformula Property

Above we have demonstrated that all cuts may be eliminated from a proof, at the possible expense of increasing the size of the proof hyperexponentially. This

normalization is worthwhile, however, since it grants one various kinds of control over the form of proofs of given sequents. One of the finest forms of control, and historically the most important, is the subformula property.

The class of *subformulas* of a given formula or sequent is defined by the following: A is a subformula of A. If A is a subformula of B, then A is also a subformula of the following formulas: $?B$, $!B$, $B \otimes C$, $C \otimes B$, $B \parr C$, $C \parr B$, $B \multimap C$, and $C \multimap B$. If A is a subformula of B, then A is also a subformula of the sequent $\vdash \Gamma_1, B, \Gamma_2$.

Corollary A.4 (Subformula Property) *Any formula in any cut-free proof of* $\vdash \Gamma$ *is a subformula of* Γ.

Proof. Each rule of linear logic except **Cut** has the property that every subformula of the hypotheses is also a subformula of the conclusion. For example, in the $\otimes$ rule, any subformula of either hypothesis is either a subformula of Σ_1, A, B, or Σ_2. However, any such subformula is also a subformula of the conclusion. In fact, we may have "added" a subformula: $(A \otimes B)$ is a subformula of the conclusion, but might not be a subformula of the hypotheses.

Therefore, by induction on the size of proofs, we have that any subformula of any step of a cut-free proof of a sequent is a subformula of the original sequent. ∎

It is easy to see that the subformula property is not true of proofs with cut: the subformulas A and $A^{\perp}$ in the hypotheses of cut might not appear in the conclusion.

Appendix B

Propositional Linear Logic Proof Rules

A linear logic sequent is a $\vdash$ followed by a multiset of linear logic formulas. Note that in standard presentations of sequent calculi, sequents are often built from sets of formulas, where we use multisets here. This difference is crucial. We assume a set of propositions p_i given, along with their associated negations, $p_i^\perp$. Below we give the inference rules for the linear sequent calculus, along with the definition of negation and implication. The reader should note that negation is a defined concept, not an operator.

The following notational conventions are followed throughout this paper:

p_i	Positive propositional literal
$p_i^\perp$	Negative propositional literal
A, B, C	Arbitrary formulas
Σ, Γ, Δ	Arbitrary multisets of formulas

Thus the identity rule (**I** below) is restricted to atomic formulas, although in fact the identity rule for arbitrary formulas ($\vdash A, A^\perp$) is derivable in this system. For notational convenience, it is usually assumed that $\multimap$ and $\otimes$ associate to the right, and that $\otimes$ has higher precedence than $\multimap$. The notation $?\Sigma$ is used to denote a multiset of formulas which all begin with ?. The English names for the rules given below are identity, cut, tensor, par, plus, with, weakening, contraction, dereliction, storage, bottom, one, and top, respectively. Note that there is no rule for the 0 constant.

I	$\dfrac{}{\vdash p_i, p_i^{\perp}}$
Cut	$\dfrac{\vdash \Sigma, A \qquad \vdash \Gamma, A^{\perp}}{\vdash \Sigma, \Gamma}$
$\otimes$	$\dfrac{\vdash \Sigma, A \qquad \vdash B, \Gamma}{\vdash \Sigma, (A \otimes B), \Gamma}$
$\parr$	$\dfrac{\vdash \Sigma, A, B}{\vdash \Sigma, (A \parr B)}$
$\oplus$	$\dfrac{\vdash \Sigma, A}{\vdash \Sigma, (A \oplus B)} \quad \dfrac{\vdash \Sigma, B}{\vdash \Sigma, (A \oplus B)}$
$\&$	$\dfrac{\vdash \Sigma, A \qquad \vdash \Sigma, B}{\vdash \Sigma, (A \,\&\, B)}$
?W	$\dfrac{\vdash \Sigma}{\vdash \Sigma, ?A}$
?C	$\dfrac{\vdash \Sigma, ?A, ?A}{\vdash \Sigma, ?A}$
?D	$\dfrac{\vdash \Sigma, A}{\vdash \Sigma, ?A}$
!S	$\dfrac{\vdash ?\Sigma, A}{\vdash ?\Sigma, !A}$
$\perp$	$\dfrac{\vdash \Sigma}{\vdash \Sigma, \perp}$
1	$\dfrac{}{\vdash 1}$
$\top$	$\dfrac{}{\vdash \Sigma, \top}$

Linear negation is defined as follows:

$$
\begin{array}{rcl}
(p_i)^\perp & \triangleq & p_i^\perp \\
(p_i^\perp)^\perp & \triangleq & p_i \\
(A \otimes B)^\perp & \triangleq & B^\perp \parr A^\perp \\
(A \parr B)^\perp & \triangleq & B^\perp \otimes A^\perp \\
(A \oplus B)^\perp & \triangleq & A^\perp \,\&\, B^\perp \\
(A \,\&\, B)^\perp & \triangleq & A^\perp \oplus B^\perp \\
(!A)^\perp & \triangleq & ?A^\perp \\
(?A)^\perp & \triangleq & !A^\perp \\
(1)^\perp & \triangleq & \perp \\
(\perp)^\perp & \triangleq & 1 \\
(0)^\perp & \triangleq & \top \\
(\top)^\perp & \triangleq & 0
\end{array}
$$

Linear implication, $\multimap$, is defined as follows:

$$A \multimap B \triangleq A^\perp \parr B$$